



Fast Models

Version 11.26

Fixed Virtual Platforms (FVP) Reference Guide

Non-Confidential

Issue 00

Copyright © 2017–2024 Arm Limited (or its affiliates). 100966_1126_00_en
All rights reserved.



Fast Models Fixed Virtual Platforms (FVP) Reference Guide

This document is Non-Confidential.

Copyright © 2017–2024 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted [Arm's Proprietary Notice](#) found at the end of this document.

This document (100966_1126_00_en) was issued on 2024-06-19. There might be a later issue at <http://developer.arm.com/documentation/100966>

The product version is 11.26.

See also: [Proprietary Notice](#) | [Product and document information](#) | [Useful resources](#)

Start Reading

If you prefer, you can skip to [the start of the content](#).

Intended audience

This document is written for software developers who are running bare metal semi-hosted applications or booting Linux on an Arm®v8 or Arm®v9 Fixed Virtual Platform (FVP).

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Contents

1. Introduction to FVPs.....	10
1.1 Types of FVP.....	10
2. Getting Started with Fixed Virtual Platforms.....	12
2.1 Requirements for FVPs.....	12
2.2 Contents of the FVP Standard Library package.....	13
2.3 FVP command-line options.....	14
2.4 Loading and running an application on an FVP.....	19
2.5 Configuring the model.....	20
2.6 FVP debug.....	21
2.7 Using the CLCD window.....	22
2.8 Timing considerations for FVPs.....	25
2.9 Ethernet with VE FVPs.....	25
2.10 Using a terminal with an FVP.....	28
2.11 Virtio P9 device component.....	31
3. Arm® CoreLink™ SGI-575 reference design FVP.....	33
3.1 About the SGI-575 FVP.....	33
3.2 SGI-575 FVP peripherals.....	33
3.2.1 Memory map for SGI-575 FVP SoC peripherals.....	34
3.2.2 Memory map for SGI-575 FVP board peripherals.....	35
3.2.3 Interrupt maps for SGI-575 FVP.....	35
4. Arm® Corstone™ SSE-300 FVP.....	37
4.1 Memory map overview for Corstone™ SSE-300.....	37
4.2 Corstone™ SSE-300 FVP peripherals.....	39
4.2.1 Corstone™ SSE-300 non-secure expansion peripherals memory map.....	40
4.2.2 Corstone™ SSE-300 secure expansion peripherals memory map.....	41
4.2.3 Corstone™ SSE-300 expansion peripheral interrupt map.....	43
4.3 Corstone™ SSE-300 peripheral protection controller expansion map.....	45
4.4 Corstone™ SSE-300 memory components.....	45
5. Arm® Corstone™ SSE-310 FVP.....	47

5.1 Corstone™ SSE-310 FVP memory map overview.....	47
5.2 Corstone™ SSE-310 FVP peripherals.....	49
5.2.1 Corstone™ SSE-310 FVP subsystem peripherals non-secure memory map.....	50
5.2.2 Corstone™ SSE-310 FVP subsystem peripherals secure memory map.....	51
5.2.3 Corstone™ SSE-310 FVP non-secure expansion peripherals memory map.....	52
5.2.4 Corstone™ SSE-310 FVP secure expansion peripherals memory map.....	54
5.2.5 Corstone SSE-310 FVP expansion peripheral interrupt map.....	55
5.3 Corstone™ SSE-310 FVP peripheral protection controller expansion map.....	57
5.4 Corstone™ SSE-310 FVP memory components.....	57
6. Arm® Corstone™-1000 FVP.....	59
6.1 Corstone™-1000 FVP modeled components.....	59
6.2 Run the Corstone™-1000 FVP with the software package.....	60
6.3 Corstone™-1000 board peripherals memory and interrupt map.....	61
6.4 Networking on Corstone™-1000 FVP.....	61
7. Juno FVP3 model.....	62
7.1 Unimplemented features and model limitations.....	62
7.2 Running the Juno FVP3 model.....	63
7.2.1 Run the Juno FVP3 model in Arm DS.....	63
7.2.2 Run the Juno FVP3 model standalone.....	63
7.2.3 Run the create_afs_image.py script.....	64
7.3 Jump-start the application clusters on bare metal.....	65
7.4 Basic setup for the TZC-400 model on bare metal.....	65
8. Arm® Neoverse™ N1 edge and Neoverse™ E1 edge reference design FVPs.....	67
8.1 About the RD-N1-E1 FVPs.....	67
8.2 Block diagrams for RD-N1-E1-edge.....	68
8.3 RD-N1-E1 FVP peripherals.....	69
8.3.1 Memory map for RD-N1-E1-edge FVP SoC peripherals.....	70
8.3.2 Memory map for RD-N1-E1-edge FVP board peripherals.....	71
8.3.3 Interrupt maps for RD-N1-E1-edge FVP.....	72
9. Arm® Neoverse™ N2 reference design FVP.....	73
9.1 About the RD-N2 FVP.....	73
9.2 RD-N2 FVP peripherals.....	74
9.2.1 Memory map for RD-N2 FVP SoC peripherals.....	74

9.2.2 Memory map for RD-N2 FVP board peripherals.....	75
9.2.3 Interrupt maps for RD-N2 FVP.....	76
10. Arm® Neoverse™ V1 reference design FVP.....	78
10.1 About the RD-V1 FVP.....	78
10.2 RD-V1 FVP peripherals.....	79
10.2.1 Memory map for RD-V1 FVP SoC peripherals.....	79
10.2.2 Memory map for RD-V1 FVP board peripherals.....	80
10.2.3 Interrupt maps for RD-V1 FVP.....	81
11. Configurable PCIe hierarchy.....	83
11.1 JSON format for the hierarchy.....	83
11.2 Common PCIe endpoint parameters.....	84
11.3 AHCI controller parameters.....	86
11.4 Host bridge parameters.....	88
11.5 SMMU test engine parameters.....	89
11.6 Root port parameters.....	91
11.7 Switch parameters.....	92
11.8 Root bridge parameters.....	92
11.9 Exerciser.....	93
11.10 Example hierarchy for a single ECAM configuration.....	95
11.11 Example hierarchy for two ECAM configuration.....	96
12. Base Platform FVPs.....	97
12.1 FVP_Base_AEMv8R.....	97
12.2 FVP_Base_AEMvA-AEMvA.....	126
12.3 FVP_Base_Cortex-A32x1.....	159
12.4 FVP_Base_Cortex-A32x2.....	187
12.5 FVP_Base_Cortex-A32x4.....	215
12.6 FVP_Base_Cortex-A34x1.....	244
12.7 FVP_Base_Cortex-A34x2.....	272
12.8 FVP_Base_Cortex-A34x4.....	300
12.9 FVP_Base_Cortex-A35x1.....	329
12.10 FVP_Base_Cortex-A35x2.....	356
12.11 FVP_Base_Cortex-A35x4.....	384
12.12 FVP_Base_Cortex-A510.....	413
12.13 FVP_Base_Cortex-A510+Cortex-A710.....	451

12.14 FVP_Base_Cortex-A520.....	482
12.15 FVP_Base_Cortex-A53x1.....	522
12.16 FVP_Base_Cortex-A53x2.....	549
12.17 FVP_Base_Cortex-A53x4.....	577
12.18 FVP_Base_Cortex-A55.....	606
12.19 FVP_Base_Cortex-A55+Cortex-A76.....	639
12.20 FVP_Base_Cortex-A57x1.....	668
12.21 FVP_Base_Cortex-A57x1-A35x1.....	695
12.22 FVP_Base_Cortex-A57x1-A53x1.....	725
12.23 FVP_Base_Cortex-A57x2.....	755
12.24 FVP_Base_Cortex-A57x2-A35x4.....	783
12.25 FVP_Base_Cortex-A57x2-A53x4.....	815
12.26 FVP_Base_Cortex-A57x4.....	847
12.27 FVP_Base_Cortex-A57x4-A35x4.....	876
12.28 FVP_Base_Cortex-A57x4-A53x4.....	910
12.29 FVP_Base_Cortex-A65.....	943
12.30 FVP_Base_Cortex-A65AE.....	978
12.31 FVP_Base_Cortex-A65AE+Cortex-A76AE.....	1013
12.32 FVP_Base_Cortex-A710.....	1044
12.33 FVP_Base_Cortex-A715.....	1082
12.34 FVP_Base_Cortex-A720.....	1120
12.35 FVP_Base_Cortex-A725.....	1161
12.36 FVP_Base_Cortex-A72x1.....	1201
12.37 FVP_Base_Cortex-A72x1-A53x1.....	1228
12.38 FVP_Base_Cortex-A72x2.....	1258
12.39 FVP_Base_Cortex-A72x2-A53x4.....	1286
12.40 FVP_Base_Cortex-A72x4.....	1318
12.41 FVP_Base_Cortex-A72x4-A53x4.....	1347
12.42 FVP_Base_Cortex-A73x1.....	1381
12.43 FVP_Base_Cortex-A73x1-A53x1.....	1408
12.44 FVP_Base_Cortex-A73x2.....	1438
12.45 FVP_Base_Cortex-A73x2-A53x4.....	1466
12.46 FVP_Base_Cortex-A73x4.....	1498
12.47 FVP_Base_Cortex-A73x4-A53x4.....	1527
12.48 FVP_Base_Cortex-A75.....	1561
12.49 FVP_Base_Cortex-A76.....	1590

12.50 FVP_Base_Cortex-A76AE.....	1620
12.51 FVP_Base_Cortex-A77.....	1649
12.52 FVP_Base_Cortex-A78.....	1679
12.53 FVP_Base_Cortex-A78AE.....	1708
12.54 FVP_Base_Cortex-A78C.....	1738
12.55 FVP_Base_Cortex-X1.....	1767
12.56 FVP_Base_Cortex-X1C.....	1797
12.57 FVP_Base_Cortex-X2.....	1826
12.58 FVP_Base_Cortex-X3.....	1864
12.59 FVP_Base_Cortex-X4.....	1903
12.60 FVP_Base_Cortex-X925.....	1943
12.61 FVP_Base_Neoverse-E1.....	1983
12.62 FVP_Base_Neoverse-N1.....	2018
12.63 FVP_Base_Neoverse-N2.....	2047
12.64 FVP_Base_Neoverse-N2x1-Neoverse-N2x1.....	2074
12.65 FVP_Base_Neoverse-N3.....	2104
12.66 FVP_Base_Neoverse-N3x1-Neoverse-N3x1.....	2131
12.67 FVP_Base_Neoverse-V1.....	2160
12.68 FVP_Base_Neoverse-V2x1-Neoverse-V2x1.....	2190
12.69 FVP_Base_Neoverse-V3.....	2219
12.70 FVP_Base_Neoverse-V3AE.....	2246
12.71 FVP_Base_Neoverse-V3AEx1-Neoverse-V3AEx1.....	2273
12.72 FVP_Base_Neoverse-V3x1-Neoverse-V3x1.....	2302
12.73 FVP_Base_RevC-2xAEMvA.....	2332
13. BaseR Platform FVPs.....	2393
13.1 FVP_BaseR_Cortex-R52Plus.....	2393
13.2 FVP_BaseR_Cortex-R52x1.....	2421
13.3 FVP_BaseR_Cortex-R52x2.....	2447
13.4 FVP_BaseR_Cortex-R52x4.....	2474
13.5 FVP_BaseR_Cortex-R82.....	2503
13.6 FVP_BaseR_Cortex-R82AE.....	2536
13.7 FVP_BaseR_Cortex-R82x1.....	2569
13.8 FVP_BaseR_Cortex-R82x2.....	2597
13.9 FVP_BaseR_Cortex-R82x4.....	2625
14. VE Platform FVPs.....	2656

14.1 FVP_VE_Cortex-A15x1.....	2656
14.2 FVP_VE_Cortex-A15x1-A7x1.....	2676
14.3 FVP_VE_Cortex-A15x2.....	2699
14.4 FVP_VE_Cortex-A15x2-A7x2.....	2720
14.5 FVP_VE_Cortex-A15x4.....	2744
14.6 FVP_VE_Cortex-A15x4-A7x4.....	2766
14.7 FVP_VE_Cortex-A17x1.....	2793
14.8 FVP_VE_Cortex-A17x1-A7x1.....	2813
14.9 FVP_VE_Cortex-A17x2.....	2836
14.10 FVP_VE_Cortex-A17x4.....	2857
14.11 FVP_VE_Cortex-A17x4-A7x4.....	2879
14.12 FVP_VE_Cortex-A5x1.....	2906
14.13 FVP_VE_Cortex-A5x2.....	2926
14.14 FVP_VE_Cortex-A5x4.....	2946
14.15 FVP_VE_Cortex-A7x1.....	2968
14.16 FVP_VE_Cortex-A7x2.....	2988
14.17 FVP_VE_Cortex-A7x4.....	3009
14.18 FVP_VE_Cortex-A9x1.....	3031
14.19 FVP_VE_Cortex-A9x2.....	3051
14.20 FVP_VE_Cortex-A9x4.....	3071
14.21 FVP_VE_Cortex-R4.....	3093
14.22 FVP_VE_Cortex-R5x1.....	3111
14.23 FVP_VE_Cortex-R5x2.....	3130
14.24 FVP_VE_Cortex-R7x1.....	3149
14.25 FVP_VE_Cortex-R7x2.....	3167
14.26 FVP_VE_Cortex-R8x1.....	3185
14.27 FVP_VE_Cortex-R8x2.....	3202
14.28 FVP_VE_Cortex-R8x4.....	3220
15. MPS2 Platform FVPs.....	3240
15.1 FVP_MPS2_AEMv8M.....	3240
15.2 FVP_MPS2_Cortex-M0.....	3277
15.3 FVP_MPS2_Cortex-M0plus.....	3311
15.4 FVP_MPS2_Cortex-M23.....	3345
15.5 FVP_MPS2_Cortex-M3.....	3380
15.6 FVP_MPS2_Cortex-M33.....	3414

15.7 FVP_MPS2_Cortex-M35P..... 3449

15.8 FVP_MPS2_Cortex-M4..... 3483

15.9 FVP_MPS2_Cortex-M52..... 3517

15.10 FVP_MPS2_Cortex-M55..... 3555

15.11 FVP_MPS2_Cortex-M7..... 3593

15.12 FVP_MPS2_Cortex-M85..... 3628

15.13 FVP_MPS2_SSE-200_Cortex-M33..... 3666

15.14 FVP_MPS2_SSE-200_Cortex-M55..... 3700

Proprietary Notice..... 3738

Product and document information..... 3740

Product status..... 3740

Revision history..... 3740

Conventions..... 3742

Useful resources..... 3744

1. Introduction to FVPs

Fixed Virtual Platforms (FVPs) enable software development without the need for real hardware.

They are available for Linux and Windows hosts, either:

- As source code examples in the Fast Models package, with the tools required to customize and build them. For more information, see the [Fast Models Reference Guide](#).
- As a set of pre-built executables in the FVP Standard Library package. This package is downloadable from [Fixed Virtual Platforms](#).

FVPs are available for all Cortex®-A, Cortex®-R, Cortex®-M, and Neoverse™ processors, and most of them support the CADI, MTI, and Iris interfaces, so can be used for debugging and for trace output.

The scope of this document is the following pre-built FVPs, which are downloadable from Arm Developer:

- FVPs that are included in the FVP Standard Library.
- Arm Architecture FVPs, downloadable from [Fixed Virtual Platforms](#).
- Arm Ecosystem FVPs, downloadable from [Arm Ecosystem FVPs](#).

1.1 Types of FVP

Most Fast Models FVPs are provided in a single license-managed library. In addition, some Architecture Envelope Model (AEM) FVPs are available for download without requiring a license.

Fast Models FVPs are based on the following platforms:

- Base Platform.
- BaseR Platform.
- Arm® Versatile™ Express development boards.
- Arm® MPS2 or Arm® MPS2+ platforms, for Cortex®-M series processors.

FVPs can be obtained in the following ways:

- License-managed FVPs, including some plug-ins and utilities, are provided in the FVP Standard Library. It is available for Windows or Linux hosts. For information on how to download it, see [Fixed Virtual Platforms](#) on Arm Developer.
- Free-of-charge AEM FVPs can be downloaded from [Arm Architecture Models](#) on Arm Developer without a license. They are available for Linux hosts only. The following FVPs are available:
 - Foundation Platform. This is a basic FVP with a minimal peripheral set. It includes a single cluster which can be configured with 1-4 AEMvA cores. It is suitable for running bare-metal

applications and for booting Linux. It supports the CADI debug interface, but does not support MTI or Iris interfaces. It is documented in the [Foundation Platform User Guide](#).

- Armv-A Base Platform RevC FVP. This is a platform model with a more extended peripheral set than the Foundation Platform. It has two clusters, each of which can be configured with 1-4 AEMvA cores. It supports Arm®v8-A architecture versions up to v8.7 and Armv9-A. It also supports CADI, MTI, and Iris debug and trace interfaces.
- Arm®v8-R AEM FVP. This FVP includes a single cluster of 1-4 AEMv8-R cores. It allows you to target AArch32 or AArch64, RAS, VFP, EL2, and other Arm®v8-R features.
- Armv-A Compliance FVP. This FVP is optimized for validating CPU implementations and can be used with the A-profile Architecture Compliance Kit (ACK) to demonstrate compliance with the Arm® architecture specification.
- Free-of-charge Arm Ecosystem FVPs, including Reference Design FVPs and the Morello Platform FVP can be downloaded from [Arm Ecosystem FVPs](#) on Arm Developer.



Arm provides validated Linux and Android deliverables for FVPs. For details, see the [Arm Development Platforms wiki](#) on Arm Community. To get started with Linux on FVPs, see [FVPs](#) on Arm Community.

Related information

[Contents of the FVP Standard Library package](#) on page 13

[Base Platform](#)

[Microcontroller Prototyping System 2](#)

[Versatile Express Model](#)

2. Getting Started with Fixed Virtual Platforms

This chapter describes how to use FVPs.

2.1 Requirements for FVPs

FVPs can run on a Windows or Linux host machine.

Host machine

Architecture

x86-64 and Arm® AArch64 host platforms are supported.

Minimum specification

At least 2GB RAM, preferably 4GB.

2GHz Intel Core2Duo, or similar, that supports the MMX, SSE, SSE2, SSE3, and SSSE3 instruction sets.

Recommended specification

At least double the RAM of the platform you intend to simulate. For example, a simulated platform containing 8GB of DRAM should be run on a 16GB host machine.

Fast Models and associated FVPs benefit most from high single-threaded performance. For example, a high frequency (4-5GHz) Intel Core i9 or i7 or AMD Ryzen 9 or 7 host CPU gives a significant improvement, between 30-60%, over Intel Xeon cores (2-3 GHz).

Operating system

Linux

Red Hat Enterprise Linux 7 or 8 (for 64-bit architectures), Ubuntu 18.04, 20.04, or 22.04 Long Term Support (LTS).

Windows

Microsoft Windows 10 64-bit.

Compiler

FVPs are built with Visual Studio 2019 and GCC 9.3.0.

Licensing

Fast Models and FVPs use either User-Based Licensing or FlexNet Licensing. Arm user-based licensing is only available to customers with a user-based licensing license. Documentation for user-based licensing is available at <https://lm.arm.com>. For assistance with user-based licensing issues, visit <https://developer.arm.com/support> and open a support case.

For FlexNet Publisher node-locked or floating licensing, the latest version of the FlexNet software is available for download from [License Server Management Software](#).

2.2 Contents of the FVP Standard Library package

The FVP Standard Library consolidates commonly-used FVPs into a single package which also contains some useful plug-ins and utilities.

The package installs:

FVPs

The FVPs are installed in the following directories:

- FVP_Base for [12. Base Platform FVPs](#) on page 97
- FVP_BaseR for [13. BaseR Platform FVPs](#) on page 2393
- FVP_MPS2 for [15. MPS2 Platform FVPs](#) on page 3240
- FVP_VE for [14. VE Platform FVPs](#) on page 2656



Note

- The FVPs are supplied as executables. The source code for these and other FVPs is available in the main Fast Models product.
- The package does not include unlicensed FVPs. These are available for download from [Arm Architecture Models](#) on Arm Developer.

Python

The `python` directory contains the standard Python libraries. These are required to run FVPs that include AVH peripherals, for example, the MPS2 Platform FVPs. To set up the environment variables for these FVPs, including automatically configuring them to use the standard Python libraries, run the command:

```
source $FVP_INSTALL_PATH/scripts/runtime.sh
```

Alternatively, manually set the `PYTHONHOME` environment variable to the location of the standard Python libraries.

Plug-ins

Fast Models plug-ins are DLLs or shared objects that provide extra functionality for FVPs, for instance trace output. To load a plug-in, pass the name of the plug-in to the FVP at startup using the `--plugin` command-line option or using the `FM_TRACE_PLUGINS` environment variable. For more information about plug-ins, see [Plug-ins for Fast Models](#) in the Fast Models Reference Guide.

Model Shell and Model Debugger

Model Shell is a command-line tool for launching FVPs. For more information about Model Shell, see [Model Shell](#) in the Fast Models Tools User Guide. Model Debugger is a symbolic debugger with a GUI that allows you to debug software running on the FVP. For more

information about Model Debugger, see [Model Debugger](#) in the Fast Models Tools User Guide. They are installed in the `bin` directory.

iris.debug Python module

`iris.debug` provides a Python scripting interface to Fast Models. It allows you to connect to and configure FVPs, perform execution control, and access registers and memory. It is installed under the `iris` directory. For more information about `iris.debug`, see [Iris Python Debug Scripting User Guide](#).

2.3 FVP command-line options

Specify these options when you launch an FVP from the command line. You can specify these options in any order.

Table 2-1: CADI or Iris-related options

Short form	Long form	Description
	<code>--iris-connect conspec</code>	<p>Start the Iris server according to the connection specification <i>conspec</i>. <i>conspec</i> is a structured string argument which can contain flags and parameters.</p> <p>Use <code>--iris-connect help</code> to print a list and description of all available Iris connection types.</p> <p>The following options are ignored when using <code>--iris-connect</code>:</p> <ul style="list-style-type: none"> <code>--iris-server</code> <code>--iris-allow-remote</code> <code>--iris-port</code> <code>--iris-port-range</code> <p>To set these connection parameters, use:</p> <pre>--iris-connect tcpserver[,port=PORT][,endport=ENDPORT][,allowRemote]</pre> <p>This command starts the TCP server on the first free port in the range <i>PORT-ENDPORT</i>, where the default for <i>PORT</i> is 7100 and the default for <i>ENDPORT</i> is <i>PORT</i> + 63. Only local connections are allowed, unless <i>allowRemote</i> is specified.</p> <p>The other supported connection type is <code>socketfd=FD</code> which uses the socket file descriptor <i>FD</i> as an established UNIX domain socket connection.</p> <p>Use <code>--iris-connect verbose=n</code> to set the logging level of the <code>IrisTcpServer</code>, where <i>n</i> is 0-3.</p>
<code>-S</code>	<code>--cadi-server</code>	Start a CADI server. This option allows a CADI-enabled debugger to connect to targets in the simulation. To shut down the server, return to the command window that you used to start the model and press Ctrl+C .
<code>-I</code>	<code>--iris-server</code>	<p>Start an Iris server. This option allows an Iris-enabled debugger to connect to targets in the simulation.</p> <p>Note: This option is deprecated. Use <code>--iris-connect</code> instead.</p>

Short form	Long form	Description
-R	--run	<p>Run the simulation immediately after the CADI or Iris server is started.</p> <p>Use this option with <code>--cadi-server</code> or <code>--iris-server</code>.</p> <p>The default is to wait until the debugger has connected before running.</p>
-L	--cadi-log	<p>Log all CADI function calls made during the simulation into XML files.</p> <p>One log file is created for each CADI target. The log files are created in the current working directory.</p> <p>The filename format is:</p> <p><code>CADIlog-<TargetInstanceName>-<ProcessID>.xml</code></p>
-i	--iris-log	<p>Log to stdout all Iris function calls that were made during the simulation.</p> <p>There are 5 possible log levels. To set a level greater than 1, specify the option multiple times, for example <code>-ii</code> for level 2.</p> <p>The log levels have the following meanings:</p> <p>0</p> <p>Logging is disabled. This value is the default.</p> <p>1</p> <p>Log messages use a compact single-line format.</p> <p>2</p> <p>Log messages use a single-line pseudo-JSON format.</p> <p>3</p> <p>Log messages use a more readable, multi-line pseudo-JSON format.</p> <p>4</p> <p>As 3 but also prints the U64JSON hex value of the message.</p> <p>Note: To set the Iris log level for all FVP invocations, use the <code>IRIS_GLOBAL_INSTANCE_LOG_MESSAGES</code> environment variable.</p>
-A	--iris-allow-remote	<p>Start an Iris server and allow connections to it from a remote workstation.</p> <p>The default is disallowed.</p> <p>Note: This option is deprecated. Use <code>--iris-connect</code> instead.</p>
-p	--print-port-number	<p>Print the port number on which the Iris or CADI server is listening.</p> <p>Use this option with <code>--iris-server</code> or <code>--cadi-server</code>.</p> <p>Tip: This option can be useful if you need to specify the port number when you connect a client to the debug server.</p>

Short form	Long form	Description
	<code>--iris-port <i>n</i></code>	<p>Set a port to use for the Iris server.</p> <p>Use this option with <code>--iris-server</code>.</p> <p>The default is 7100.</p> <p>Note: This option is deprecated. Use <code>--iris-connect</code> instead.</p>
	<code>--iris-port-range <i>min:max</i></code>	<p>Set the range of ports to scan when starting an Iris server. The server uses the first available port in the range.</p> <p>Use this option with <code>--iris-server</code>.</p> <p>Note: This option is deprecated. Use <code>--iris-connect</code> instead.</p>

Table 2-2: Output-related options

Short form	Long form	Description
	<code>--list-instances</code>	<p>Print a list of model instances to standard output, then exit the simulation.</p> <p>Use this option to help identify the correct syntax for configuration files, and to find out what instances the target supplies.</p>
<code>-l</code>	<code>--list-params</code>	<p>Print a list of all available model parameters and their values to standard output, then exit the simulation.</p> <p>Tip: If you are loading a plug-in, this option also lists the plug-in parameters.</p>
	<code>--list-set-params <i>filename</i></code>	<p>Save a list of the parameter values that are set for each component to a file. If <i>filename</i> is <code>-</code>, print to standard output instead.</p> <p>An alternative to this command-line option is to use the <code>FASTSIM_LOGPARAMS</code> and <code>FASTSIM_LOGPARAMS_FILE</code> environment variables:</p> <ul style="list-style-type: none"> To log parameter values to stdout, set <code>FASTSIM_LOGPARAMS</code> to 1. To log parameter values to a file, additionally set <code>FASTSIM_LOGPARAMS_FILE</code> to the name of the file.
	<code>--dump-params</code>	Dump the list of model parameters and their values into a JSON file called <code>parameter_list.json</code> , then exit the simulation. The file is created in the current working directory.
	<code>--list-regs</code>	Print model register information to standard output, then exit the simulation.
	<code>--check-regs</code>	Same as <code>--list-regs</code> but with extra consistency checks on the CADI register API.
<code>-o</code>	<code>--output <i>filename</i></code>	<p>Redirect output from the <code>--list-instances</code>, <code>--list-memory</code>, <code>--list-params</code>, and <code>--list-regs</code> commands to a file.</p> <p>If this option is used with <code>--list-params</code>, the contents of the output file are formatted correctly for use as input by the <code>--config-file</code> option.</p>
	<code>--log <i>filename</i></code>	Log all SystemC reports into <i>filename</i> .

Short form	Long form	Description
	<code>--stat</code>	<p>Print the following performance statistics on simulation exit:</p> <p>Simulated time An estimate of the time that the workload would have taken on the modeled hardware.</p> <p>User time Time in wall clock seconds that the host CPU spent running in user mode.</p> <p>System time Time in wall clock seconds that the host CPU spent running in system mode.</p> <p>Wall time Time in wall clock seconds between the simulation starting and stopping.</p> <p>Performance index An estimate of the accuracy of the simulation performance. This value is Simulated time divided by Wall time.</p>
<code>-P</code>	<code>--prefix</code>	Prefix each line of semihosting output with the name of the target instance.
<code>-h</code>	<code>--help</code>	Print the help message and exit.
	<code>--version</code>	Print version information for the FVP.
<code>-q</code>	<code>--quiet</code>	Suppress informational output.
<code>-K</code>	<code>--keep-console</code>	Keep the console window open after completion. This option applies to Windows hosts only.
	<code>--disable-model-exitcode</code>	Disable the simulation from retrieving the exit code returned by a model or a plug-in. By default, it is enabled.

Table 2-3: Run control options

Short form	Long form	Description
	<code>--cpulimit <i>n</i></code>	<p>Maximum number of wall-clock seconds for the simulation process to be active. This value excludes simulation startup and shutdown.</p> <p>This option is ignored if a debug server is started.</p> <p>The default is unlimited.</p>
	<code>--cyclelimit <i>n</i></code>	<p>Maximum number of cycles to run.</p> <p>This option is ignored if a debug server is started.</p> <p>The default is unlimited.</p>
<code>-T</code>	<code>--timelimit <i>n</i></code>	Maximum number of wall-clock seconds for the simulation to run, excluding startup and shutdown. To terminate the model immediately after initialization, specify <code>--timelimit 0</code> .
	<code>--simlimit <i>n</i></code>	<p>Maximum number of seconds to simulate.</p> <p>This option is ignored if a debug server is started.</p> <p>The default is unlimited.</p> <p>Like the <code>Simulated time</code> value output by <code>--stat</code>, this value is measured in simulation seconds, not wall-clock seconds.</p>

Short form	Long form	Description
-b	<code>--break [instance=] [threadid:][memspace@] address</code>	<p>Set a program breakpoint on the address of an instruction, where:</p> <ul style="list-style-type: none"> threadid is an optional thread id, for a thread-specific breakpoint. memspace is an optional name or id of the memory space that address is in. If not specified, the breakpoint is set on the first program memory space found. If the FVP has multiple cores, you must specify an instance, for example: <pre>-b FVP_Base_AEMvA.cluster0.cpu0=0x800010eC</pre> <p>This option can be specified multiple times.</p>

Table 2-4: Timing and performance options

Short form	Long form	Description
	<code>--cpi-file filename</code>	<p>Use <i>filename</i> to set the Cycles Per Instruction (CPI) class.</p> <p>For information about CPI files, see Timing Annotation.</p>
-Q	<code>--quantum n</code>	Number of ticks to simulate for each quantum. The default is 10000.
-M	<code>--min-sync- latency n</code>	Number of ticks to simulate before synchronizing. Events that occur at a higher frequency than this value are missed. The default is 100.
	<code>--fast-ram filename</code>	Enable FastRAM and load the configuration from <i>filename</i> . For more information about FastRAM, see FastRAM .

Table 2-5: Memory-related options

Short form	Long form	Description
	<code>--dump file@address,size</code>	<p>Dump a section of memory to a file at model shutdown. This option can be specified multiple times. The full syntax is:</p> <pre>--dump [instance=] file@[memspace:]address,size</pre> <p>Tip: To see the list of instances and memory spaces, use the <code>--list-memory</code> option.</p>
	<code>--data file@address</code>	<p>Write raw data contained in <i>file</i> to the specified address. This option can be specified multiple times. The full syntax is:</p> <pre>--data [instance=] file@[memspace:]address</pre>
	<code>--list-memory</code>	Print model memory information to standard output, then exit the simulation.
	<code>--start [instance=] address</code>	<p>Set the initial PC value to this address, overriding the <code>.axf</code> start address.</p> <p>Note:</p> <ul style="list-style-type: none"> Use this option if you do not want the CPU to start executing at the default reset address. You do not normally need to do this if you are loading an ELF file using <code>--application</code>. This option can be used with <code>--data</code> to load binary data that is not in an ELF file.

Table 2-6: Configuration options

Short form	Long form	Description
-C	<code>--parameter <i>instance.parameter=value</i></code>	Set a parameter. This option can be specified multiple times. Specify the full hierarchical name of the parameter. This option is also used to set plug-in parameters.
-f	<code>--config-file <i>filename</i></code>	Load the parameters from a configuration file.

Table 2-7: Options for loading a plug-in or application

Short form	Long form	Description
-a	<code>--application [<i>instance=</i>] <i>filename</i></code>	Load an application. On a multi-core system, specify the instance, or use * to load the application image into all the cores in a cluster: <code>-a cluster0.cpu*=<i>file</i></code>
	<code>--plugin <i>filename</i></code>	Load the plug-in <i>filename</i> . This option can be specified multiple times. You can also load plug-ins using the FM_TRACE_PLUGINS environment variable. For information about plug-ins, see Plug-ins for Fast Models .
	<code>--trace-plugin <i>filename</i></code>	Load a trace plug-in. Note: This option is deprecated. Use <code>--plugin</code> instead.

2.4 Loading and running an application on an FVP

There are different ways to launch an FVP, for example from the command prompt, or from Model Debugger or Arm® Development Studio.

To run an FVP from the command prompt, enter the model name followed by the model options. To see all available options, use the `--help` option. This is a list of some of the commonly used options for FVPs:

-a [*instance=*]*filename.axf*

Specifies an application to load, and optionally, the instance or instances to load it on. The file can be in one of the following formats, or in a gzip-compressed version:

- ELF.
- Motorola S-Record.

If the FVP contains multiple core instances, you can specify the instance to load the image on. The instance name can include a wildcard (*) to load the same application image into multiple cores, for example:

```
./FVP_Base_AEMvA -a cluster0.cpu*=__image.axf
```

Omitting the instance name loads the application on all cores in the first cluster. If the FVP has multiple cores but no clusters, you must specify the instance name.

--data *filename.bin@address*

Loads binary data into memory at the address specified.

-C *instance.parameter=value*

Sets a single model parameter. Parameters are specified using a path that separates the instance names and the parameter using dots. For example, `-c bp.flashloader0.fname=fip.bin`. Here, `bp` and `flashloader0` are instance names and `fname` is the parameter. To set multiple parameters using a configuration file, use the `-f` option instead. To list all the available parameters, with their type, default value, and description, run the model with the `--list-params`, or `-l` option.

-f *config_file.txt*

Specifies the name of a plain text configuration file. Configuration files simplify managing multiple model parameters. You can set the same parameters using this option as with the `-c` option.

-S

Starts a CADI debug server. This option allows a CADI-enabled debugger, such as Model Debugger or Arm® Development Studio Debugger, to connect to the running model. By default, the model waits for the debugger to connect before starting.

Related information

[Arm Development Studio Getting Started Guide](#)
[Model Debugger](#)

2.5 Configuring the model

When you start the model from the command line, you can configure it using either:

- One or more `-c` command-line arguments.
- A configuration file and the `-f` command-line argument.

Each `-c` command-line argument or line in the configuration file must contain:

- The name of the component instance.
- The parameter to modify.
- Its value.

Use the following format:

instance.parameter=value

The *instance* can be a hierarchical path, with each level separated by a dot . character.



- Comment lines in the configuration file begin with a # character.
- You can set Boolean values using either `true` or `false`, or 1 or 0.

You can generate a configuration file with all parameters set to default values by using the `-o` option to redirect the output from the `--list-params` option, for example:

```
FVP_Base_AEMvA.exe --list-params -o params.txt
```

2.6 FVP debug

This section describes how to debug an FVP.

FVP debug options

To debug an FVP, you can either:

- Run the FVP from within a CADI-enabled debugger.
- Start the FVP with the `-s` command-line argument and then connect a CADI-enabled debugger to it.

For information about using your debugger in these ways, see your debugger documentation.

Semihosting support

Semihosting enables code running on a platform model to directly access the I/O facilities on a host computer. Examples of these facilities include console I/O and file I/O.

The simulator handles semihosting by intercepting `HLT 0xF000`, `svc 0x123456`, or `svc 0xAB`, depending on whether the processor is in A64, A32 or T32 state. It handles all other `HLTs` and `svcs` as normal.

If the operating system does not use `HLT 0xF000`, `svc 0x123456`, or `svc 0xAB` for its own purposes, it is not necessary to disable semihosting support to boot an operating system.

To temporarily or permanently disable semihosting support for a current debug connection, see your debugger documentation.

Related information

[Semihosting for AArch32 and AArch64](#)

[Using semihosting to access resources on the host computer](#)

2.7 Using the CLCD window

When an FVP starts, the CLCD window opens, representing the contents of the simulated color LCD frame buffer. It automatically resizes to match the horizontal and vertical resolution that is set in the CLCD peripheral registers.

MPS2 FVPs

The LEDs and MCC switches in the CLCD window for MPS2-based FVPs correspond to the LEDs and switches on the physical board. They are controlled by the software that you load onto the FVP. For information on how to use them, see [User switches and user LEDs](#) in the Arm® MPS2 and MPS2+ FPGA Prototyping Boards TRM on Arm Developer.

Base Platform and VE FVPs

The top section of the CLCD window for Base Platform and VE FVPs displays the status information:

Figure 2-1: Base Platform CLCD window in its default state at startup



USERSW

Eight white boxes show the state of the User DIP switches.

These represent switch S6 on the VE hardware, USERSW[8:1], which is mapped to bits [7:0] of the SYS_SW register at address 0x1C010004.

The switches are in the off position by default. To change its state, click in the area above or below a white box.

BOOTSW

Eight white boxes show the state of the VE Boot DIP switches.

These represent switch S8 on the VE hardware, BOOTSEL[8:1], which is mapped to bits [15:8] of the SYS_SW register at address 0x1C010004.

The switches are in the off position by default.



Changing Boot DIP switch positions while the model is running can result in unpredictable behavior.

S6LED

Eight colored boxes indicate the state of the VE User LEDs.

These represent the red/yellow/green LEDs on the VE hardware, which are mapped to bits [7:0] of the SYS_LED register at address 0x1C010008.

Daughter

Eight white boxes show the state of the daughterboard DIP switches and eight colored boxes show the state of the daughterboard LEDs.

Total Instr

A counter showing the total number of instructions executed.

Because the FVP models provide a *Programmer's View* (PV) of the system, the CLCD displays total instructions rather than total processor cycles. Timing might differ substantially from the hardware because:

- Bus fabric is simplified.
- Memory latencies are minimized.
- Cycle approximate processor and peripheral models are used.

In general, bus transaction timing is consistent with the hardware, but the timing of operations within the model is not accurate.

Total Time

A counter showing the total elapsed time, in seconds.

This time is wall clock time, not simulated time.

Rate Limit

A feature that disables or enables fast simulation.

Because the system model is highly optimized, your code might run faster than it would on real hardware. This effect might cause timing issues.

Rate Limit is disabled by default to favor simulation speed. Enable it to restrict simulation time so that it more closely matches real time.

To enable or disable Rate Limit, click the square button. You can also configure this option when instantiating the model with the `rate_limit-enable` visualization component parameter.

When you click the **Total Instr** item in the CLCD, the display toggles to show the following:

Instr/sec

The number of instructions that execute per second of wall clock time.

Perf Index

The ratio of real time to simulation time. The larger the ratio, the faster the simulation runs. If you enable the Rate Limit feature, the Perf Index approaches unity.

You can reset the simulation counters by resetting the model.

The FVP CLCD displays the core run state for each core on each cluster using a colored icon. The icons are to the left of the **Total Instr** (or **Instr/sec**) item.

Figure 2-2: Core run state icons for a dual-cluster, quad-core model



Table 2-8: Core run state icon descriptions

Icon	State label	Description
	UNKNOWN	Run status unknown, that is, simulation has not started.
	RUNNING	Core running, is not idle, and is executing instructions.
	HALTED	External halt signal asserted.
	STANDBY_WFE	Last instruction executed was WFE and standby mode has been entered.
	STANDBY_WFI	Last instruction executed was WFI and standby mode has been entered.
	IN_RESET	External reset signal asserted.
	DORMANT	Partial core power down.
	SHUTDOWN	Complete core power down.

If the CLCD window has focus:

- Any keyboard input is translated to PS/2 keyboard data.
- Any mouse activity over the window is translated into PS/2 relative mouse motion data. The data is then streamed to the KMI peripheral model FIFOs.



The simulator only sends relative mouse motion events to the model. As a result, the host mouse pointer does not necessarily align with the target OS mouse pointer.

You can hide the host mouse pointer by pressing the **left Ctrl+left Alt** keys. Press the keys again to redisplay the host mouse pointer. Only the **left Ctrl** key is operational. The **right Ctrl** key does not have the same effect.

If you prefer to use a different key, configure it with the `trap_key` visualization component parameter.

Related information

[VEVisualisation component](#)

2.8 Timing considerations for FVPs

The Rate Limit feature matches simulation time to wall clock time.

FVPs provide an environment for running software applications in a functionally-accurate simulation. However, because they prioritize simulation speed over timing accuracy, there are situations where they might behave unexpectedly.

When code interacts with real world devices like timers and keyboards, data arrives in the modeled device in real world, or wall clock, time, but simulation time can run much faster than the wall clock. This means that a single key press might register as several repeated key presses, or a single mouse click incorrectly becomes a double click.

Enabling Rate Limit, either using the Rate Limit button in the CLCD display, or the `rate_limit-enable` model instantiation parameter, forces the model to run at wall clock time. This avoids issues with two clocks running at significantly different rates. Rate Limit is disabled by default. For interactive applications, Arm recommends enabling it.

2.9 Ethernet with VE FVPs

This section describes how to use Ethernet with VE FVPs.

Using Ethernet with VE FVPs

The VE FVPs have a virtual Ethernet component. This component is a model of the SMSC 91C111 Ethernet controller, and uses a TAP device to communicate with the network. By default, the Ethernet component is disabled.

Host requirements

Before you can use the Ethernet capability of VE FVPs, set up your host computer.

Target requirements

This section describes the target requirements.

Target requirements - about

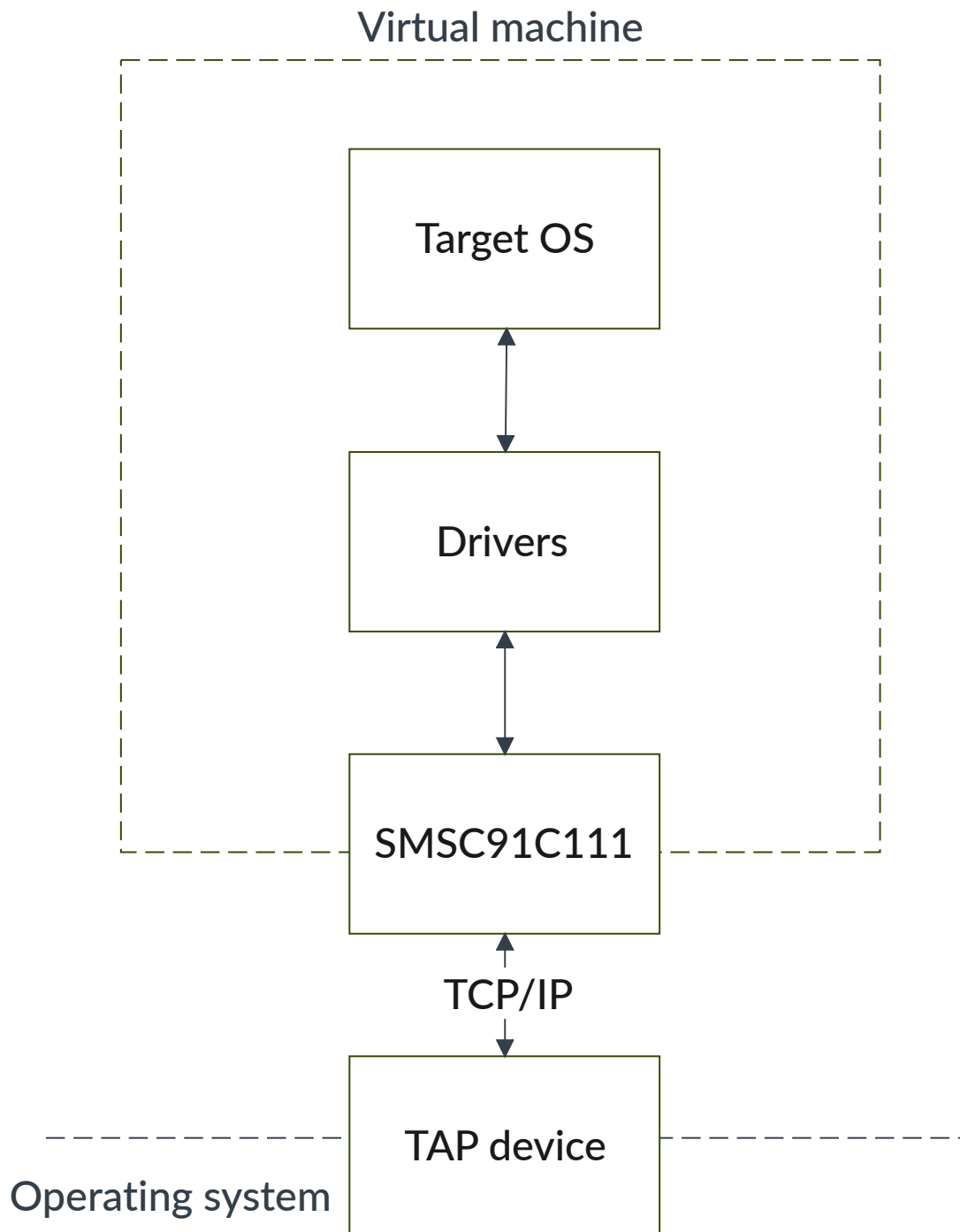
The VE FVPs include a software implementation of the SMSC 91C111 Ethernet controller. Your target OS must therefore include a driver for this specific device. To use the SMSC chip, configure the kernel. Linux supports the SMSC 91C111.

The configurable SMSC 91C111 component parameters are:

- `enabled`.
- `mac_address`.
- `promiscuous`.

enabled

When the device is disabled, the kernel cannot detect the device.

Figure 2-3: Model networking structure block diagram

To perform read and write operations on the TAP device, configure a HostBridge component. The HostBridge component is a virtual *Programmer's View* (PV) model. It acts as a networking

gateway to exchange Ethernet packets with the TAP device on the host, and to forward packets to NIC models.

mac_address

There are two options for the `mac_address` parameter.

If a MAC address is not specified, when the simulator is run it takes the default MAC address, which is randomly generated. This random generation provides some degree of MAC address uniqueness when running models on multiple hosts on a local network.

promiscuous

The Ethernet component starts in promiscuous mode by default. In this mode, it receives all network traffic, even any not addressed to the device. Use this mode if you are using a single network device for multiple MAC addresses. Use this mode if, for example, you share the network card between your host OS and the VE FVP Ethernet component.

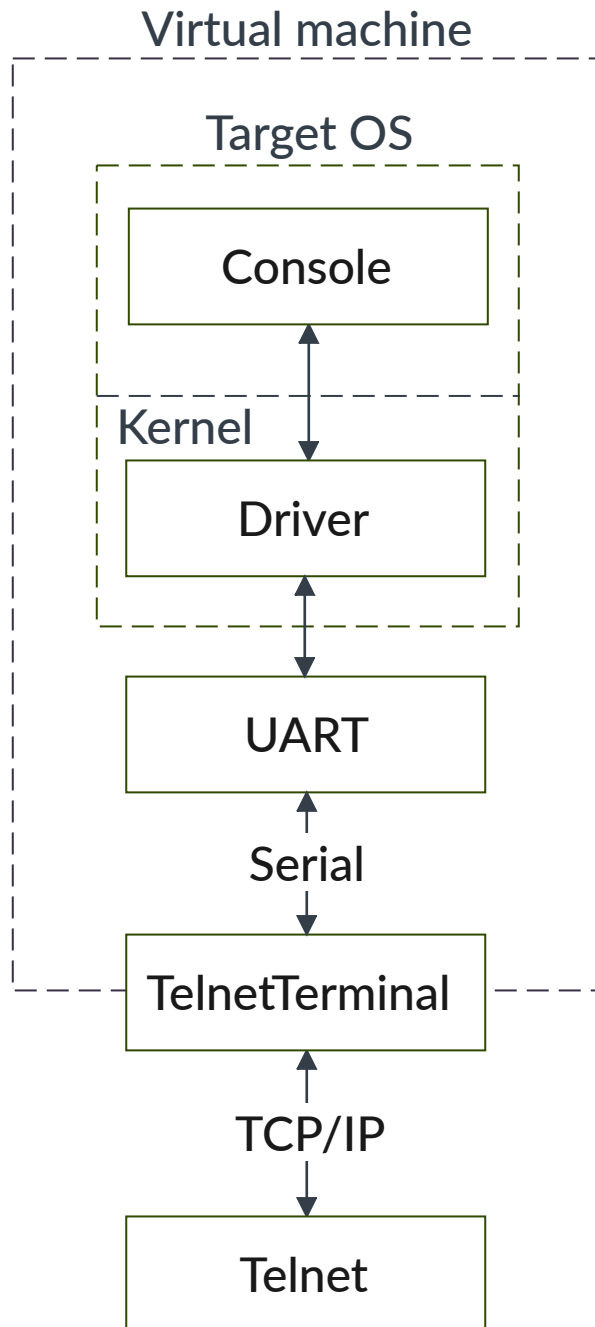
By default, the Ethernet device on the VE FVP has a randomly generated MAC address and starts in promiscuous mode.

2.10 Using a terminal with an FVP

The `TelnetTerminal` component is a virtual component that enables UART data to be transferred between a TCP/IP socket on the host and a serial port on the target.

Using TelnetTerminal

The following figure shows a block diagram of one possible relationship between the target and host through the `TelnetTerminal` component. The `TelnetTerminal` block is what you configure when you define Terminal component parameters. The Virtual Machine is an FVP.

Figure 2-4: Terminal block diagram

On the target side, the console process that is invoked by your target OS relies on a suitable driver being present. Such drivers are normally part of the OS kernel. The driver passes serial data

through a UART. The data is forwarded to the TelnetTerminal component. When the simulation is started and the TelnetTerminal component is enabled, the component opens a server (listening) socket on a TCP/IP port. This is port 5000 by default. This port can be connected to by, for example, a Telnet process on the host.

When data becomes available on the network socket, the TelnetTerminal component buffers the data, which can then be read from SerialData.

If there is no connection to the network socket when the first data access is made, and the `start_telnet` parameter is true, a host Telnet session is started automatically. Prior to this first access, you can connect a client of your choice to the network socket.

If the connection between the TelnetTerminal component and the client is broken at any time, for example by closing a client Telnet session, the port is re-opened on the host, permitting you to make another client connection. This could have a different port number if the original one is no longer available.

The port number of a particular TelnetTerminal instance can be defined when your model system starts. The actual value of the port used by each TelnetTerminal is declared when it starts or restarts, and might not be the value that you specified if the port is already in use. If you are using Model Shell, the port numbers are displayed in the host window in which you started the model.



Note

Microsoft Windows 10 disables the Telnet client by default. Follow these steps to enable it:

1. Select **Start > Settings**.
2. In the search box, type **Turn Windows features on or off**. The **Windows Features** dialog opens.
3. Select the **Telnet Client** check box and click **OK**. The installation might take several minutes to complete.

TelnetTerminal parameters

To set the parameters, the syntax to use in a configuration file or on the command line is:

```
motherboard.terminal_x.parameter=value
```

where *x* is the terminal identifier and can be 0, 1, 2, or 3.

You can start the TelnetTerminal component in either of the following modes, depending on the `mode` parameter:

telnet

In Telnet mode, the terminal component supports a subset of the RFC 854 protocol. This means that the terminal participates in negotiations between the host and client concerning what is and is not supported, but there is no flow control.

raw

In raw mode the byte stream passes unmodified between the host and the target. The terminal does not participate in initial capability negotiations between the host and client. Instead it acts as a TCP/IP port. You can use this feature to directly connect to your target through the TelnetTerminal component. This permits a debugger connection, for example, to connect a gdb client to a gdbserver running on the target operating system.

The `terminal_command` parameter specifies the command line used to launch a terminal application and connect to the opened TCP port. The TelnetTerminal component replaces the keywords `%port` and `%title`, if specified, with the opened port number and component name, respectively. After replacing `%port` and `%title`, the command line is executed verbatim.

An empty string, which is the default, launches `xterm` on Linux or `telnet.exe` on Windows.



If you specify a non-empty string, it must include `%port`, but `%title` is optional.

For example:

```
fvp_mps2.telnetterminal0.terminal_command="putty.exe -telnet localhost %port"
```

Related information

[TelnetTerminal](#)

2.11 Virtio P9 device component

The VirtioP9Device component is included in Base, BaseR, and A-profile VE platforms. It implements a subset of the Plan 9 file protocol over a virtio transport. It enables accessing a directory on the host's filesystem within Linux, or another operating system that implements the protocol, running on a platform model.

Setting up the VirtioP9Device component

Take the following steps to set up this component:

- Use a version of Linux that supports v9fs over virtio and virtio-mmio devices.
- Update the device tree to include the VirtioP9Device component, or specify it on the kernel command-line, as shown below. The address range for both VE and Base platforms is `0x1C140000-0x1C14FFFF`.
The interrupt number is 43, or IRQ 75, for both VE and Base platforms.
- Set the following parameter to the directory on the host that you want to mount in the model:

VE:

```
motherboard.virtiop9device.root_path
```

Base:

```
bp.virtio_p9device.root_path
```

- On Linux, mount the host directory by using the following command in the model:

```
$ mount -t 9p -o trans=virtio,version=9p2000.L FM <mount point>
```

Example kernel command-line argument:

```
virtio_mmio.device=0x10000@0x1c140000:75
```

Example entry for DTS files, to add next to the corresponding `virtio_block` entry:

```
virtio_p9@0140000 {  
    compatible = "virtio,mmio";  
    reg = <0x0 0x1c140000 0x0 0x1000>;  
    interrupts = <0x0 0x2b 0x4>;  
};
```

3. Arm® CoreLink™ SGI-575 reference design FVP

The SGI-575 FVP provides a software model of the Arm® CoreLink™ SGI-575 System Guidance for Infrastructure reference subsystem. It drives system architecture and software standardization and provides software and binaries of proprietary firmware that reduce the amount of work that is required for SoC development.

The FVP is used with the SGI-575 software package.

See the Arm® CoreLink™ SGI-575 System Guidance for Infrastructure Software Bundle Readme for instructions on how to set up and run the FVP.

3.1 About the SGI-575 FVP

The SGI-575 Fixed Virtual Platform (FVP) models multiple IP components.



It does not model every component that SGI-575 describes. For example, it does not model the Arm® CoreSight™ technology components.

The FVP models the following IP components:

- 2 × MP4 Cortex®-A75 clusters
- SCP
- MCP
- CMN-600
- Multiple NIC-400 interconnects
- Memory access path towards DRAM

3.2 SGI-575 FVP peripherals

The SGI-575 FVP includes peripherals that the software payload requires to run.

These peripherals are organized in two layers:

SoC

The SoC peripherals represent peripherals that are added to a compute subsystem in a SoC design.

Board

The board peripherals represent peripherals that are present on the board onto which the SoC is mounted.

3.2.1 Memory map for SGI-575 FVP SoC peripherals

This table shows the memory map for the SoC peripherals in the SGI-575 FVP.

Table 3-1: SoC peripherals

Name	Base address	Size	Description
SMC interface	0x00_0800_0000	368MB	Routed to Board
DMA MMU-400	0x00_7FB0_0000	64KB	-
HDLCD1 MMU-400	0x00_7FB1_0000	64KB	-
HDLCD0 MMU-400	0x00_7FB2_0000	64KB	-
SoC Interconnect NIC400 GPV	0x00_7FD0_0000	1MB	-
Surge detector	0x00_7FE5_0000	4KB	Dummy APB
TRNG	0x00_7FE6_0000	4KB	-
Trusted Non-volatile counters	0x00_7FE7_0000	4KB	-
Trusted Root-Key storage	0x00_7FE8_0000	4KB	-
Secure I2C	0x00_7FE9_0000	256B	PL061 GPIO
DDR4 PHY 1	0x00_7FB7_0000	64KB	Dummy APB
DDR4 PHY 0	0x00_7FB6_0000	64KB	Dummy APB
DMA Non-secure	0x00_7FF0_0000	4KB	-
DMA Secure	0x00_7FF1_0000	4KB	-
HDCLD1	0x00_7FF5_0000	4KB	-
HDCLD1	0x00_7FF6_0000	4KB	-
UART 1	0x00_7FF7_0000	4KB	-
UART 0	0x00_7FF8_0000	4KB	-
I2S	0x00_7FF9_0000	1KB	PL061 GPIO
I2C	0x00_7FFA_0000	256B	PL061 GPIO
PL352	0x00_7FFD_0000	4KB	PL354
System override Registers	0x00_7FFF_0000	4KB	-
AP configuration	0x00_7FFE_0000	4KB	GPR

3.2.2 Memory map for SGI-575 FVP board peripherals

This table shows the memory map for the board peripherals in the SGI-575 FVP.

Table 3-2: Board peripherals

Name	Base address	Size	Description
NOR Flash 0	0x00_0800_0000	64MB	-
NOR Flash 1	0x00_0C00_0000	64MB	-
NOR Flash 2	0x00_1000_0000	64MB	-
Ethernet	0x00_1800_0000	64MB	SMSC 91C111
System Registers	0x00_1C01_0000	64KB	-
MCI	0x00_1C05_0000	64KB	PL180
KMI 0	0x00_1C06_0000	64KB	PL050
KMI 1	0x00_1C07_0000	64KB	PL050
UART 0	0x00_1C09_0000	64KB	PL011
UART 1	0x00_1C0A_0000	64KB	PL011
VFS2	0x00_1C0D_0000	64KB	-
Watchdog	0x00_1C0F_0000	64KB	SP805
Dual Timer	0x00_1C11_0000	64KB	SP084
Virtio Block Device	0x00_1C13_0000	64KB	-
RTC	0x00_1C17_0000	64KB	PL031
GPIO 0	0x00_1C1D_0000	64KB	-
GPIO 1	0x00_1C1E_0000	64KB	-
DRAM	0x00_8000_0000	2GB	-
DRAM	0x80_8000_0000	6GB	-

3.2.3 Interrupt maps for SGI-575 FVP

These tables show the interrupt IDs and sources for the FVP peripherals.

Table 3-3: Interrupt map at the SoC layer

Interrupt ID	Source	Description
147	UART 0	-
148	UART 1	-
171	TRNG	-

Table 3-4: Interrupt map at the board layer

Interrupt ID	Source	Description
111	Ethernet	-
132	RTC	-
133	UART 0	-

Interrupt ID	Source	Description
134	UART 1	-
140	VFS2	-
202	Virtio	-
228	Watchdog	-
229	KMI 0	-
230	Dual Timer	Interrupts 0 and 1
231	System Registers Ethernet IRQ	-

4. Arm® Corstone™ SSE-300 FVP

To develop ahead of hardware availability and to explore the design from a software perspective, the Corstone™ SSE-300 Fixed Virtual Platform (FVP) models much of the Arm® IP in the Corstone™ SSE-300 subsystem version r0p1.

The Corstone™ SSE-300 FVP drives system architecture and software standardization. It is used with the Corstone™ SSE-300 software package which provides software and binaries of proprietary firmware that reduce the amount of work that is required for SoC development. See the Corstone™ SSE-300 software bundle readme for instructions on how to set up and run the FVP.

The FVP models the following IP components:

- A single Arm® Cortex®-M55 processor with the MVE extension.
- A single Arm® Ethos™-U55 or Ethos™-U65 processor.
- Memory Protection Controller (MPC).
- Peripheral Protection Controller (PPC).
- Implementation Defined Attribution Unit (IDAU).

The full description of components that are internal to the SSE-300 subsystem can be found in the [Arm Corstone SSE-300 Example Subsystem Technical Reference Manual](#).

The FVP has the following limitations:

- It does not model every component that Corstone™ SSE-300 describes. For example, it does not model the CoreSight™ technology components.
- QSPI is not modeled. It has QSPI SRAM instead.
- Some components are dummy stubs with a minimal implementation. Models with dummy APB have memory access.
- Some components are partially modeled.

Refer to the following documents for more information:

- [Arm Corstone SSE-300 Example Subsystem Technical Reference Manual](#)
- [Arm MPS3 FPGA Prototyping Board Technical Reference Manual](#)
- [Corstone-300 on Arm Developer](#)

4.1 Memory map overview for Corstone™ SSE-300

This table outlines the main FVP memories and their positions within the memory map.

This memory map includes IDAU security information for memory regions.

Table 4-1: Memory map overview

Row ID	Address range		Size	Description	Alias with Row ID	IDAU region values		
	From	To				Security	IDAUID	NSC
1	0x0000_0000	0x0007_FFFF	512KB	ITCM ³	5	NS	0	0
2	0x0008_0000	0x00FF_FFFF	15.5MB	Reserved	-			
3	0x0100_0000	0x010F_FFFF	1MB	SRAM (only 1MB) ¹	7			
4	0x0110_0000	0x0FFF_FFFF	239MB	Reserved	-			
5	0x1000_0000	0x1007_FFFF	512KB	ITCM ³	1	S	1	CODE NSC
6	0x1008_0000	0x10FF_FFFF	15.5MB	Reserved	-			
7	0x1100_0000	0x110F_FFFF	1MB	SRAM (only 1MB) ¹	3			
8	0x1110_0000	0x1FFF_FFFF	239MB	Reserved	-			
9	0x2000_0000	0x2007_FFFF	512KB	DTCM (4 x banks of 128KB) ³	15	NS	2	0
10	0x2008_0000	0x20FF_FFFF	15.5MB	Reserved	-	-	-	-
11	0x2100_0000	0x211F_FFFF	2MB	Internal SRAM area (SSE-300 implements 2x1MB) ³	17	-	-	-
12	0x2120_0000	0x27FF_FFFF	110MB	Reserved	-	-	-	-
13	0x2800_0000	0x287F_FFFF	8MB	QSPI (only 8MB) ¹	19	-	-	-
14	0x2880_0000	0x2FFF_FFFF	120MB	Reserved	-	-	-	-
15	0x3000_0000	0x3007_FFFF	512KB	DTCM (4 x banks of 128KB) ³	9	S	3	RAM NSC
16	0x3008_0000	0x30FF_FFFF	15.5MB	Reserved	-	-	-	-
17	0x3100_0000	0x311F_FFFF	2MB	Internal SRAM area (SSE-300 implements 2x1MB) ³	11	-	-	-
18	0x3120_0000	0x37FF_FFFF	110MB	Reserved	-	-	-	-
19	0x3800_0000	0x387F_FFFF	8MB	QSPI (only 8MB) ¹	13	-	-	-
20	0x3880_0000	0x3FFF_FFFF	120MB	Reserved	-	-	-	-
21	0x4000_0000	0x47FF_FFFF	128MB	Non-Secure Low Latency Peripheral Region.	23	NS	4	0
22	0x4800_0000	0x4FFF_FFFF	128MB	Non-Secure High Latency Peripheral Region.	24	NS	4	0
23	0x5000_0000	0x57FF_FFFF	128MB	Secure Low Latency Peripheral Region.	21	S	5	0
24	0x5800_0000	0x5FFF_FFFF	128MB	Secure High Latency Peripheral Region.	22	S	5	0
25	0x6000_0000	0x6FFF_FFFF	256MB	DDR4 ¹	-	NS	6	0
26	0x7000_0000	0x7FFF_FFFF	256MB	DDR4 ¹	-	S	7	0
27	0x8000_0000	0x8FFF_FFFF	256MB	DDR4 ¹	-	NS	8	0
28	0x9000_0000	0x9FFF_FFFF	256MB	DDR4 ¹	-	S	9	0
29	0xA000_0000	0xAFFF_FFFF	256MB	DDR4 ¹	-	NS	A	0
30	0xB000_0000	0xBFFF_FFFF	256MB	DDR4 ¹	-	S	B	0
31	0xC000_0000	0xCFFF_FFFF	256MB	DDR4 ¹	-	NS	C	0

Row ID	Address range		Size	Description	Alias with Row ID	IDAU region values		
	From	To				Security	IDAUID	NSC
32	0xD000_0000	0xDFFF_FFFF	256MB	DDR4 ¹	-	S	D	0
33	0xE000_0000	0xE00F_FFFF	1MB	External Private Peripheral Bus	-	-	Exempt	-
34	0xE010_0000	0xE01F_FFFF	1MB	Reserved	-	NS	E	0
35	0xE020_0000	0xEFFF_FFFF	254MB	Maps to HMSTEXPPILL expansion interface ²	-	NS	E	0
36	0xF000_0000	0xF00F_FFFF	1MB	Reserved	-	-	Exempt	-
37	0xF010_0000	0xF01F_FFFF	1MB	Reserved	-	S	F	0
38	0xF020_0000	0xFFFF_FFFF	254MB	Maps to HMSTEXPPILL expansion interface ²	-	S	F	0

4.2 Corstone™ SSE-300 FVP peripherals

The Corstone™ SSE-300 FVP includes peripherals that the software payload requires to run.

Board peripherals are peripherals that might be present on the board onto which the SoC is mounted. The Corstone™ SSE-300 board model is based on the Arm® MPS3 board.



Some of the MPS3 Fast Models have minimal implementations. For more information, refer to the documents in the FVP package.

All peripherals that are extensions to the Corstone™ SSE-300 subsystem are mapped into two key areas of the memory map:

0x4000_0000 to 0x47FF_FFFF and 0x4800_0000 to 0x4FFF_FFFF

Non-secure region that maps to AHB Master Expansion 1 interface.

0x5000_0000 to 0x57FF_FFFF and 0x5800_0000 to 0x5FFF_FFFF

Secure region that maps to AHB Master Expansion 1 interface.

¹ Security access is controlled by MPC.

² Accesses to these addresses result in an AHB5 error response.

³ For security settings, control, and features, refer to the Arm® Corstone™ SSE-300 documentation.

4.2.1 Corstone™ SSE-300 non-secure expansion peripherals memory map

Memory map for non-secure board peripherals.

Table 4-2: Non-secure board peripherals

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
1	0x4110_0000	0x4110_0FFF	4KB	GPIO 0	CMSDK GPIO
2	0x4110_1000	0x4110_1FFF	4KB	GPIO 1	CMSDK GPIO
3	0x4110_2000	0x4110_2FFF	4KB	GPIO 2	CMSDK GPIO
4	0x4110_3000	0x4110_3FFF	4KB	GPIO 3	CMSDK GPIO
5	0x4110_4000	0x4110_4FFF	4KB	FMC GPIO 0	CMSDK GPIO
6	0x4110_5000	0x4110_5FFF	4KB	FMC GPIO 1	CMSDK GPIO
7	0x4110_6000	0x4110_6FFF	4KB	FMC GPIO 2	CMSDK GPIO
8	0x4110_7000	0x4110_7FFF	4KB	FMC USER AHB	Not modeled
-	0x4110_8000	0x411F_FFFF	-	Reserved	-
9	0x4120_0000	0x4120_0FFF	4KB	USER AHB 0	Not modeled
10	0x4120_1000	0x4120_1FFF	4KB	USER AHB 1	Not modeled
11	0x4120_2000	0x4120_2FFF	4KB	USER AHB 2	Not modeled
12	0x4120_3000	0x4120_3FFF	4KB	USER AHB 3	Not modeled
-	0x4120_4000	0x413F_FFFF	-	Reserved	-
13	0x4140_0000	0x414F_FFFF	1MB	Ethernet	SMSC91C111 Ethernet controller
14	0x4150_0000	0x415F_FFFF	1MB	USB	Dummy stub
-	0x4160_0000	0x416F_FFFF	-	Reserved	-
15	0x4170_0000	0x4170_0FFF	4KB	User APB0	Not modeled
16	0x4170_1000	0x4170_1FFF	4KB	User APB1	Not modeled
17	0x4170_2000	0x4170_2FFF	4KB	User APB2	Not modeled
18	0x4170_3000	0x4170_3FFF	4KB	User APB3	Not modeled
-	0x4170_4000	0x417F_FFFF	-	Reserved	-
19	0x4180_0000	0x4180_0FFF	4KB	QSPI Config	Not modeled
20	0x4180_1000	0x4180_1FFF	4KB	QSPI Write	Not modeled
-	0x4180_2000	0x47FF_FFFF	-	Reserved	-
-	0x4800_0000	0x480F_FFFF	-	Subsystem peripherals	-
21	0x4810_2000	0x4810_2FFF	4KB	Ethos-U55 APB	Modeled
22	0x4810_3000	0x4810_31FF	0.5KB	U55 timing adapter 0 APB	Modeled
23	0x4810_3200	0x4810_33FF	0.5KB	U55 timing adapter 1 APB	Modeled
-	0x4810_3400	0x491F_FFFF	-	Reserved	-

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
24	0x4920_0000	0x4920_0FFF	4KB	FPGA - SBCon I2C (Touch)	Partially modeled
25	0x4920_1000	0x4920_1FFF	4KB	FPGA - SBCon I2C (Audio Conf)	Dummy stub
26	0x4920_2000	0x4920_2FFF	4KB	FPGA - PL022 (SPI ADC)	Dummy stub
27	0x4920_3000	0x4920_3FFF	4KB	FPGA - PL022 (SPI Shield0)	Dummy stub
28	0x4920_4000	0x4920_4FFF	4KB	FPGA - PL022 (SPI Shield1)	Dummy stub
29	0x4920_5000	0x4920_5FFF	4KB	SBCon (I2C - Shield0)	Dummy stub
30	0x4920_6000	0x4920_6FFF	4KB	SBCon (I2C - Shield1)	Dummy stub
31	0x4920_7000	0x4920_7FFF	4KB	USER APB	Dummy stub
32	0x4920_8000	0x4920_8FFF	4KB	FPGA - SBCon I2C (DDR4 EEPROM)	Dummy stub
-	0x4920_9000	0x492F_FFFF	-	Reserved	-
33	0x4930_0000	0x4930_0FFF	4KB	FPGA - SCC registers	Modeled
34	0x4930_1000	0x4930_1FFF	4KB	FPGA - I2S (Audio)	Partially modeled
35	0x4930_2000	0x4930_2FFF	4KB	FPGA - IO (System Ctrl + I/O)	Modeled
36	0x4930_3000	0x4930_3FFF	4KB	UART0 - UART_F [0]	CMSDK UART
37	0x4930_4000	0x4930_4FFF	4KB	UART1 - UART_F [1]	CMSDK UART
38	0x4930_5000	0x4930_5FFF	4KB	UART2 - UART_F [2]	CMSDK UART
39	0x4930_6000	0x4930_6FFF	4KB	UART3 - UART Shield 0	Dummy stub
40	0x4930_7000	0x4930_7FFF	4KB	UART4 - UART Shield 1	Dummy stub
41	0x4930_8000	0x4930_8FFF	4KB	UART5 - UART_F [3]	CMSDK UART
42	0x4930_9000	0x4930_9FFF	4KB	Reserved	
43	0x4930_A000	0x4930_AFFF	4KB	CLCD Config Reg	Partially modeled
44	0x4930_B000	0x4930_BFFF	4KB	RTC	PL031_RTC

4.2.2 Corstone™ SSE-300 secure expansion peripherals memory map

Memory map for secure board peripherals.

Table 4-3: Secure board peripherals

Row ID	Address range in secure region		Size	Description	Modeled in FVP
	From	To			
1	0x5110_0000	0x5110_0FFF	4KB	GPIO 0	CMSDK GPIO

Row ID	Address range in secure region		Size	Description	Modeled in FVP
	From	To			
2	0x5110_1000	0x5110_1FFF	4KB	GPIO 1	CMSDK GPIO
3	0x5110_2000	0x5110_2FFF	4KB	GPIO 2	CMSDK GPIO
4	0x5110_3000	0x5110_3FFF	4KB	GPIO 3	CMSDK GPIO
5	0x5110_4000	0x5110_4FFF	4KB	FMC GPIO 0	CMSDK GPIO
6	0x5110_5000	0x5110_5FFF	4KB	FMC GPIO 1	CMSDK GPIO
7	0x5110_6000	0x5110_6FFF	4KB	FMC GPIO 2	CMSDK GPIO
8	0x5110_7000	0x5110_7FFF	4KB	FMC USER AHB	Not modeled
	0x5110_8000	0x511F_FFFF		Reserved	
9	0x5120_0000	0x5120_0FFF	4KB	USER AHB 0	Not modeled
10	0x5120_1000	0x5120_1FFF	4KB	USER AHB 1	Not modeled
11	0x5120_2000	0x5120_2FFF	4KB	USER AHB 2	Not modeled
12	0x5120_3000	0x5120_3FFF	4KB	USER AHB 3	Not modeled
	0x5120_4000	0x513F_FFFF		Reserved	
13	0x5140_0000	0x514F_FFFF	1MB	Ethernet	SMSC91C111 Ethernet controller
14	0x5150_0000	0x515F_FFFF	1MB	USB	Dummy stub
	0x5160_0000	0x516F_FFFF		Reserved	
15	0x5170_0000	0x5170_0FFF	4KB	User APB0	Not modeled
16	0x5170_1000	0x5170_1FFF	4KB	User APB1	Not modeled
17	0x5170_2000	0x5170_2FFF	4KB	User APB2	Not modeled
18	0x5170_3000	0x5170_3FFF	4KB	User APB3	Not modeled
	0x5170_4000	0x517F_FFFF		Reserved	
19	0x5180_0000	0x5180_0FFF	4KB	QSPI Config	Not modeled
20	0x5180_1000	0x5180_1FFF	4KB	QSPI Write	Not modeled
	0x5180_2000	0x56FF_FFFF		Reserved	
21	0x5700_0000	0x5700_0FFF	4KB	SRAM Memory Protection Controller (MPC)	Modeled
22	0x5700_1000	0x5700_1FFF	4KB	QSPI Memory Protection Controller (MPC)	Modeled
23	0x5700_2000	0x5700_2FFF	4KB	DDR4 Memory Protection Controller (MPC)	Modeled
	0x5700_3000	0x57FF_FFFF		Reserved	
	0x5800_0000	0x580F_FFFF		Subsystem peripherals	
24	0x5810_2000	0x5810_2FFF	4KB	Ethos-U55 APB	Modeled
25	0x5810_3000	0x5810_31FF	0.5KB	U55 timing adapter 0 APB	Modeled

Row ID	Address range in secure region		Size	Description	Modeled in FVP
	From	To			
26	0x5810_3200	0x5810_33FF	0.5KB	U55 timing adapter 1 APB	Modeled
	0x5810_3400	0x591F_FFFF		Reserved	
27	0x5920_4000	0x5920_4FFF	4KB	FPGA - PL022 (SPI Shield1)	Dummy stub
28	0x5920_5000	0x5920_5FFF	4KB	SBCon (I2C - Shield0)	Dummy stub
29	0x5920_6000	0x5920_6FFF	4KB	SBCon (I2C - Shield1)	Dummy stub
30	0x5920_7000	0x5920_7FFF	4KB	USER APB	Dummy stub
31	0x5920_8000	0x5920_8FFF	4KB	DDR4 EEPROM	Dummy stub
	0x5920_9000	0x592F_FFFF		Reserved	
32	0x5930_0000	0x5930_0FFF	4KB	FPGA - SCC registers	Modeled
33	0x5930_1000	0x5930_1FFF	4KB	FPGA - I2S (Audio)	Partially modeled
34	0x5930_2000	0x5930_2FFF	4KB	FPGA - IO (System Ctrl + I/O)	Modeled
35	0x5930_3000	0x5930_3FFF	4KB	UART0 - UART_F [0]	CMSDK UART
36	0x5930_4000	0x5930_4FFF	4KB	UART1 - UART_F [1]	CMSDK UART
37	0x5930_5000	0x5930_5FFF	4KB	UART2 - UART_F [2]	CMSDK UART
38	0x5930_6000	0x5930_6FFF	4KB	UART3 - UART Shield 0	Dummy stub
39	0x5930_7000	0x5930_7FFF	4KB	UART4 - UART Shield 1	Dummy stub
40	0x5930_8000	0x5930_8FFF	4KB	UART5 - UART_F [3]	CMSDK UART
	0x5930_9000	0x5930_9FFF	4KB	Reserved	
41	0x5930_A000	0x5930_AFFF	4KB	CLCD Config Reg	Partially modeled
42	0x5930_B000	0x5930_BFFF	4KB	RTC	PL031_RTC

4.2.3 Corstone™ SSE-300 expansion peripheral interrupt map

Interrupt map at the board layer.

Table 4-4: Interrupt map at the board layer

Interrupt input	Interrupt source
IRQ [32]	System timestamp counter interrupt. Not implemented.
IRQ [33]	UART 0 receive interrupt.
IRQ [34]	UART 0 transmit Interrupt.
IRQ [35]	UART 1 receive interrupt.
IRQ [36]	UART 1 transmit interrupt.
IRQ [37]	UART 2 receive interrupt.
IRQ [38]	UART 2 transmit interrupt.

Interrupt input	Interrupt source
IRQ [39]	UART 3 receive interrupt.
IRQ [40]	UART 3 transmit interrupt.
IRQ [41]	UART 4 receive interrupt.
IRQ [42]	UART 4 transmit interrupt.
IRQ [43]	UART 0 combined interrupt.
IRQ [44]	UART 1 combined interrupt.
IRQ [45]	UART 2 combined interrupt.
IRQ [46]	UART 3 combined interrupt.
IRQ [47]	UART 4 combined interrupt.
IRQ [48]	UART overflow (0, 1, 2, 3, 4, and 5).
IRQ [49]	Ethernet.
IRQ [50]	Audio I2S.
IRQ [51]	Touch screen.
IRQ [52]	USB.
IRQ [53]	SPI ADC.
IRQ [54]	SPI (shield 0).
IRQ [55]	SPI (shield 1).
IRQ [56]	Ethos-U interrupt.
IRQ [68:57]	Reserved.
IRQ [69]	GPIO 0 combined interrupt.
IRQ [70]	GPIO 1 combined interrupt.
IRQ [71]	GPIO 2 combined interrupt.
IRQ [72]	GPIO 3 combined interrupt.
IRQ [88:73]	GPIO 0 individual interrupts.
IRQ [104:89]	GPIO 1 individual interrupts.
IRQ [120:105]	GPIO 2 individual interrupts.
IRQ [124:121]	GPIO 3 individual interrupts.
IRQ [125]	UART 5 receive interrupt.
IRQ [126]	UART 5 transmit interrupt.
IRQ [127]	UART 5 combined interrupt.
IRQ [130:128]	Reserved.

4.3 Corstone™ SSE-300 peripheral protection controller expansion map

The Corstone™ SSE-300 FVP implements secure access configuration registers which control security and privileged accesses to peripherals connected to PPC.

Table 4-5: Secure access configuration registers - PPC bits

Bit	MAIN_PPCEXP0 (AHB0)	MAIN_PPCEXP1 (AHB1)	PERIPH_PPCEXP0 (APB0)	PERIPH_PPCEXP1 (APB1)	PERIPH_PPCEXP2 (APB2)
0	GPIO-0	User AHB 0	-	SBCon I2C (touch screen)	FPGA - SCC registers
1	GPIO-1	User AHB 1	-	SBCon I2C (audio conf)	FPGA - I2S (audio)
2	GPIO-2	User AHB 2	-	FPGA PL022 (SPI2 for ADC)	FPGA - GPIO (System Ctrl + I/O)
3	GPIO-3	User AHB 3	-	FPGA PL022 (SPI Shield0)	UART0
4	FMC GPIO-0	-	NPU APB0	FPGA PL022 (SPI Shield1)	UART1
5	FMC GPIO-1	-	NPU APB1	FPGA SBCon (I2C - Shield0)	UART2
6	FMC GPIO-2	-	-	FPGA SBCon (I2C - Shield1)	UART3 STUB
7	FMC User AHB	-	-	Reserved	UART4 STUB
8	USB and Ethernet	-	-	FPGA - SBCon I2C (DDR4 EPROM)	UART5
9	-	-	-	-	Reserved
10	-	-	-	-	CLCD config reg
11	-	-	-	-	RTC
12	-	-	-	-	-
13	-	-	SRAM MPC	-	-
14	-	-	QSPI MPC	-	-
15	-	-	DDR MPC	-	-
PPC IRQ no.	5	6	2	3	4

4.4 Corstone™ SSE-300 memory components

The Corstone™ SSE-300 FVP includes the following memory components and their security access is controlled by the MPC.

Table 4-6: Memory components

Name	Non-secure address range	Secure alias	Size
SRAM	0x0100_0000 - 0x010F_FFFF	0x1100_0000 - 0x110F_FFFF	1MB
QSPI SRAM	0x2800_0000 - 0x287F_FFFF	0x3800_0000 - 0x387F_FFFF	8MB
DDR0	0x6000_0000 - 0x6FFF_FFFF	0x7000_0000 - 0x7FFF_FFFF	256MB
DDR1	0x8000_0000 - 0x8FFF_FFFF	0x9000_0000 - 0x9FFF_FFFF	256MB

Name	Non-secure address range	Secure alias	Size
DDR2	0xA000_0000 - 0xAFFF_FFFF	0xB000_0000 - 0xBFFF_FFFF	256MB
DDR3	0xC000_0000 - 0xCFFF_FFFF	0xD000_0000 - 0xDFFF_FFFF	256MB

5. Arm® Corstone™ SSE-310 FVP

To develop ahead of hardware availability and explore the design from a software perspective, the Corstone™ SSE-310 Fixed Virtual Platform (FVP) models much of the Arm® IP in the Corstone™ SSE-310 design. It models version r0p0 of the Corstone™ SSE-310 subsystem.

The FVP drives system architecture and software standardization. It is used with the Corstone™ SSE-310 software package which provides software and binaries of proprietary firmware that reduce the amount of work that is required for SoC development. See the Corstone™ SSE-310 software bundle readme for instructions on how to set up and run the FVP.

The FVP models the following IP components:

- A single Arm® Cortex®-M85 processor with MVE extension.
- A single Arm® Ethos™-U55 processor.
- Memory Protection Controller (MPC).
- Peripheral Protection Controller (PPC).
- Implementation Defined Attribution Unit (IDAU).

The full description of components that are internal to the SSE-310 subsystem can be found in the SSE-310 Example Subsystem TRM.

The FVP has the following limitations:

- It does not model every component that Corstone™ SSE-310 describes. For example, it does not model the CoreSight™ technology components.
- QSPI is not modeled. It has QSPI SRAM instead.
- It has a single DMA-350 engine instead of four PL081 DMA peripherals.
- Some components are dummy stubs with a minimal implementation. Models with dummy APB have memory access.
- Some components are partially modeled.

Refer to the following documents for more information:

- [Arm Corstone SSE-310 Example Subsystem Technical Reference Manual](#)
- [Arm MPS3 FPGA Prototyping Board Technical Reference Manual](#)
- [Corstone-310 on Arm Developer](#)

5.1 Corstone™ SSE-310 FVP memory map overview

This table outlines the main FVP memories and their positions within the memory map.

This memory map includes IDAU security information for memory regions:

Table 5-1: Memory map overview

Row ID	Address range		Size	Description	Alias with Row ID	IDAU region values		
	From	To				Security	IDAUID	NSC
1	0x0000_0000	0x0000_7FFF	32KB	ITCM ⁶	5	NS	0	0
2	0x0000_8000	0x00FF_FFFF	15.9MB	Reserved	-			
3	0x0100_0000	0x011F_FFFF	2MB	SRAM (2MB) ⁴	7			
4	0x0120_0000	0x0FFF_FFFF	238MB	Reserved	-			
5	0x1000_0000	0x1000_7FFF	32KB	ITCM ⁶	1	S	1	CODE NSC
6	0x1000_8000	0x10FF_FFFF	15.9MB	Reserved	-			
7	0x1100_0000	0x111F_FFFF	2MB	SRAM (2MB) ⁴	3			
8	0x1120_0000	0x1FFF_FFFF	238MB	Reserved	-			
9	0x2000_0000	0x2000_7FFF	32KB	DTCM (4 x banks of 8KB) ⁶	15	NS	2	0
10	0x2000_8000	0x20FF_FFFF	15.9MB	Reserved	-	-	-	-
11	0x2100_0000	0x213F_FFFF	4MB	Internal SRAM area (SSE-310 implements 2x2MB) ⁶	17	-	-	-
12	0x2140_0000	0x27FF_FFFF	108MB	Reserved	-	-	-	-
13	0x2800_0000	0x287F_FFFF	8MB	QSPI (only 8MB) ⁴	19	-	-	-
14	0x2880_0000	0x2FFF_FFFF	120MB	Reserved	-	-	-	-
15	0x3000_0000	0x3000_7FFF	32KB	DTCM (4 x banks of 8KB) ⁶	9	S	3	RAM NSC
16	0x3000_8000	0x30FF_FFFF	15.9MB	Reserved	-	-	-	-
17	0x3100_0000	0x313F_FFFF	4MB	Internal SRAM area (SSE-310 implements 2x2MB) ⁶	11	-	-	-
18	0x3140_0000	0x37FF_FFFF	108MB	Reserved	-	-	-	-
19	0x3800_0000	0x387F_FFFF	8MB	QSPI (only 8MB) ⁴	13	-	-	-
20	0x3880_0000	0x3FFF_FFFF	120MB	Reserved	-	-	-	-
21	0x4000_0000	0x47FF_FFFF	128MB	Non-Secure Low Latency Peripheral Region.	23	NS	4	0
22	0x4800_0000	0x4FFF_FFFF	128MB	Non-Secure High Latency Peripheral Region.	24	NS	4	0
23	0x5000_0000	0x57FF_FFFF	128MB	Secure Low Latency Peripheral Region.	21	S	5	0
24	0x5800_0000	0x5FFF_FFFF	128MB	Secure High Latency Peripheral Region.	22	S	5	0
25	0x6000_0000	0x6FFF_FFFF	256MB	DDR4 ⁴	-	NS	6	0
26	0x7000_0000	0x7FFF_FFFF	256MB	DDR4 ⁴	-	S	7	0
27	0x8000_0000	0x8FFF_FFFF	256MB	DDR4 ⁴	-	NS	8	0
28	0x9000_0000	0x9FFF_FFFF	256MB	DDR4 ⁴	-	S	9	0
29	0xA000_0000	0xAFFF_FFFF	256MB	DDR4 ⁴	-	NS	A	0
30	0xB000_0000	0xBFFF_FFFF	256MB	DDR4 ⁴	-	S	B	0
31	0xC000_0000	0xCFFF_FFFF	256MB	DDR4 ⁴	-	NS	C	0

Row ID	Address range		Size	Description	Alias with Row ID	IDAU region values		
	From	To				Security	IDAUID	NSC
32	0xD000_0000	0xDFFF_FFFF	256MB	DDR4 ⁴	-	S	D	0
33	0xE000_0000	0xE00F_FFFF	1MB	External Private Peripheral Bus	-	-	Exempt	-
34	0xE010_0000	0xE01F_FFFF	1MB	Reserved	-	NS	E	0
35	0xE020_0000	0xEFFF_FFFF	254MB	Maps to HMSTEXPPILL expansion interface ⁵	-	NS	E	0
36	0xF000_0000	0xF00F_FFFF	1MB	Reserved	-	-	Exempt	-
37	0xF010_0000	0xF01F_FFFF	1MB	Reserved	-	S	F	0
38	0xF020_0000	0xFFFF_FFFF	254MB	Maps to HMSTEXPPILL expansion interface ⁵	-	S	F	0

5.2 Corstone™ SSE-310 FVP peripherals

The Corstone™ SSE-310 FVP includes peripherals that the software payload requires to run.

These peripherals are organized in the following layers:

Subsystem

The subsystem peripherals represent peripherals that are present on the SoC.

Board

The board peripherals represent peripherals that may be present on the board onto which the SoC is mounted. The Corstone™ SSE-310 board model is based on the Arm® MPS3 board.

All peripherals that are extensions to the Corstone™ SSE-310 subsystem are mapped into two key areas of the memory map:

0x4000_0000 to 0x47FF_FFFF and 0x4800_0000 to 0x4FFF_FFFF

Non-Secure region that maps to the AHB Master Expansion 1 interface.

0x5000_0000 to 0x57FF_FFFF and 0x5800_0000 to 0x5FFF_FFFF

Secure region that maps to the AHB Master Expansion 1 interface.



Some of the MPS3 Fast Models have minimal implementations. For more information, refer to the documents in the FVP package.

⁴ Security access is controlled by MPC.

⁵ Accesses to these addresses result in an AHB5 error response.

⁶ For security settings, control, and features, refer to the Arm® Corstone™ SSE-310 documentation.

5.2.1 Corstone™ SSE-310 FVP subsystem peripherals non-secure memory map

Non-secure memory map for subsystem peripherals.

Table 5-2: Non-secure subsystem peripherals

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
1	0x4000_0000	0x4000_0FFF	4KB	MHU 0	Not modeled
2	0x4000_1000	0x4000_1FFF	4KB	MHU 1	Not modeled
3	0x4000_2000	0x4000_3FFF	8KB	DMA	DMA-350
4	0x4000_4000	0x4000_4FFF	4KB	NPU APB	Modeled
-	0x4000_5000	0x4001_EFFF	-	Reserved	-
5	0x4001_F000	0x4001_FFFF	4KB	CPUID	Modeled
-	0x4002_0000	0x4007_0000	-	Reserved	-
6	0x4008_0000	0x4008_0FFF	4KB	Non-secure Access Configuration Registers	Modeled
-	0x4008_1000	0x4008_FFFF	-	Reserved	-
7	0x4009_0000	0x4009_3FFF	16KB	CryptoCell312	Not modeled
-	0x4009_4000	0x400F_FFFF	-	Reserved	-
-	0x4010_0000	0x47FF_FFFF	-	Board peripherals	-
8	0x4800_0000	0x4800_0FFF	4KB	Timer 0	Modeled
9	0x4800_1000	0x4800_1FFF	4KB	Timer 1	Modeled
10	0x4800_2000	0x4800_2FFF	4KB	Timer 2	Modeled
11	0x4800_3000	0x4800_3FFF	4KB	Timer 3	Modeled
-	0x4800_4000	0x4801_FFFF	-	Reserved	-
12	0x4802_0000	0x4802_0FFF	4KB	SYSINFO	Modeled
-	0x4802_1FFF	0x4802_EFFF	-	Reserved	-
13	0x4802_F000	0x4802_FFFF	4KB	SLOWCLK Timer	Modeled
-	0x4803_0000	0x4803_FFFF	-	Reserved	-
14	0x4804_0000	0x4804_0FFF	4KB	Non-Secure Watchdog Control Frame	Modeled
15	0x4804_1000	0x4804_1FFF	4KB	Non-Secure Watchdog Refresh Frame	Modeled
-	0x4804_2000	0x480F_FFFF	-	Reserved	-

5.2.2 Corstone™ SSE-310 FVP subsystem peripherals secure memory map

Secure memory map for subsystem peripherals.

Table 5-3: Secure subsystem peripherals

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
1	0x5000_0000	0x5000_0FFF	4KB	MHU 0	Not modeled
2	0x5000_1000	0x5000_1FFF	4KB	MHU 1	Not modeled
3	0x5000_2000	0x5000_3FFF	8KB	DMA	DMA-350
4	0x5000_4000	0x5000_4FFF	4KB	NPU APB	Modeled
-	0x5000_5000	0x5001_EFFF	-	Reserved	-
5	0x5001_F000	0x5001_FFFF	4KB	CPUID	Modeled
-	0x5002_0000	0x5007_0000	-	Reserved	-
6	0x5008_0000	0x5008_0FFF	4KB	Secure Access Configuration Registers	Modeled
-	0x5008_1000	0x5008_2FFF	-	Reserved	-
7	0x5008_3000	0x5008_3FFF	4KB	Internal SRAM MPC 0	Modeled
8	0x5008_4000	0x5008_4FFF	4KB	Internal SRAM MPC 1	Modeled
-	0x5008_5000	0x5008_FFFF	-	Reserved	-
9	0x5009_0000	0x5009_3FFF	16KB	CryptoCell312	Not modeled
-	0x5009_4000	0x500F_FFFF	-	Reserved	-
-	0x5010_0000	0x57FF_FFFF	-	Board peripherals	-
10	0x5800_0000	0x5800_0FFF	4KB	Timer 0	Modeled
11	0x5800_1000	0x5800_1FFF	4KB	Timer 1	Modeled
12	0x5800_2000	0x5800_2FFF	4KB	Timer 2	Modeled
13	0x5800_3000	0x5800_3FFF	4KB	Timer 3	Modeled
-	0x5800_4000	0x5801_FFFF	-	Reserved	-
14	0x5802_0000	0x5802_0FFF	4KB	SYSINFO	Modeled
15	0x5802_1000	0x5802_1FFF	4KB	SYSCONTROL	Modeled
16	0x5802_2000	0x5802_2FFF	4KB	SYS_PPU	Modeled
17	0x5802_3000	0x5802_3FFF	4KB	CPU0_PPU	Modeled
-	0x5802_4000	0x5802_6FFF	-	Reserved	-
18	0x5802_7000	0x5802_7FFF	4KB	CRYPTO_PPU	Not modeled
19	0x5802_8000	0x5802_8FFF	4KB	MGMT_PPU	Modeled
20	0x5802_9000	0x5802_9FFF	4KB	DBG_PPU	Modeled
21	0x5802_A000	0x5802_AFFF	4KB	NPU_PPU	Modeled
-	0x5802_B000	0x5802_DFFF	-	Reserved	-
22	0x5802_E000	0x5802_EFFF	4KB	SLOWCLK Watchdog	Modeled

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
23	0x5802_F000	0x5802_FFFF	4KB	SLOWCLK Timer	Modeled
-	0x5803_0000	0x5803_FFFF	-	Reserved	-
24	0x5804_0000	0x5804_0FFF	4KB	Secure Watchdog Control Frame	Modeled
25	0x5804_1000	0x5804_1FFF	4KB	Secure Watchdog Refresh Frame	Modeled
-	0x5804_2000	0x580F_FFFF	-	Reserved	-

5.2.3 Corstone™ SSE-310 FVP non-secure expansion peripherals memory map

Non-secure memory map for board peripherals.

Table 5-4: Non-secure board peripherals

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
1	0x4110_0000	0x4110_0FFF	4KB	GPIO 0	CMSDK GPIO
2	0x4110_1000	0x4110_1FFF	4KB	GPIO 1	CMSDK GPIO
3	0x4110_2000	0x4110_2FFF	4KB	GPIO 2	CMSDK GPIO
4	0x4110_3000	0x4110_3FFF	4KB	GPIO 3	CMSDK GPIO
5	0x4110_4000	0x4110_4FFF	4KB	USER AHB 0	Not modeled
6	0x4110_5000	0x4110_5FFF	4KB	USER AHB 1	Not modeled
7	0x4110_6000	0x4110_6FFF	4KB	USER AHB 2	Not modeled
8	0x4110_7000	0x4110_7FFF	4KB	USER AHB 3	Not modeled
-	0x4110_8000	0x413F_FFFF	-	Reserved	-
9	0x4140_0000	0x414F_FFFF	1MB	Ethernet	SMSC91C111 Ethernet controller
10	0x4150_0000	0x415F_FFFF	1MB	USB	Dummy stub
-	0x4160_0000	0x416F_FFFF	-	Reserved	-
11	0x4170_0000	0x4170_0FFF	4KB	Timing adapter APB0 - FPGA SRAM	Dummy stub
12	0x4170_1000	0x4170_1FFF	4KB	Timing adapter APB1 - QSPI	Dummy stub
13	0x4170_2000	0x4170_2FFF	4KB	Timing adapter APB2 - DDR4	Dummy stub
14	0x4170_3000	0x4170_3FFF	4KB	User APB3	Not modeled
-	0x4170_4000	0x417F_FFFF	-	Reserved	-
15	0x4180_0000	0x4180_0FFF	4KB	QSPI Config	Not modeled
16	0x4180_1000	0x4180_1FFF	4KB	QSPI Write	Not modeled
-	0x4180_2000	0x47FF_FFFF	-	Reserved	-

Row ID	Address range in non-secure region		Size	Description	Modeled in FVP
	From	To			
-	0x4800_0000	0x480F_FFFF	-	Subsystem peripherals	-
-	0x4810_0000	0x491F_FFFF	-	Reserved	-
17	0x4920_0000	0x4920_0FFF	4KB	FPGA - SBCon I2C (Touch)	Partially modeled
18	0x4920_1000	0x4920_1FFF	4KB	FPGA - SBCon I2C (Audio Conf)	Dummy stub
19	0x4920_2000	0x4920_2FFF	4KB	FPGA - PL022 (SPI ADC)	Dummy stub
20	0x4920_3000	0x4920_3FFF	4KB	FPGA - PL022 (SPI Shield0)	Dummy stub
21	0x4920_4000	0x4920_4FFF	4KB	FPGA - PL022 (SPI Shield1)	Dummy stub
22	0x4920_5000	0x4920_5FFF	4KB	SBCon (I2C - Shield0)	Dummy stub
23	0x4920_6000	0x4920_6FFF	4KB	SBCon (I2C - Shield1)	Dummy stub
24	0x4920_7000	0x4920_7FFF	4KB	USER APB	Dummy stub
25	0x4920_8000	0x4920_8FFF	4KB	FPGA - SBCon I2C (DDR4 EEPROM)	Dummy stub
-	0x4920_9000	0x492F_FFFF	-	Reserved	-
26	0x4930_0000	0x4930_0FFF	4KB	FPGA - SCC registers	Modeled
27	0x4930_1000	0x4930_1FFF	4KB	FPGA - I2S (Audio)	Partially modeled
28	0x4930_2000	0x4930_2FFF	4KB	FPGA - IO (System Ctrl + I/O)	Modeled
29	0x4930_3000	0x4930_3FFF	4KB	UART0 - UART_F [0]	CMSDK UART
30	0x4930_4000	0x4930_4FFF	4KB	UART1 - UART_F [1]	CMSDK UART
31	0x4930_5000	0x4930_5FFF	4KB	UART2 - UART_F [2]	CMSDK UART
32	0x4930_6000	0x4930_6FFF	4KB	UART3 - UART Shield 0	Dummy stub
33	0x4930_7000	0x4930_7FFF	4KB	UART4 - UART Shield 1	Dummy stub
34	0x4930_8000	0x4930_8FFF	4KB	UART5 - UART_F [3]	CMSDK UART
-	0x4930_9000	0x4930_9FFF	4KB	Reserved	-
35	0x4930_A000	0x4930_AFFF	4KB	CLCD Config Reg	Partially modeled
36	0x4930_B000	0x4930_BFFF	4KB	RTC	PL031_RTC

5.2.4 Corstone™ SSE-310 FVP secure expansion peripherals memory map

Secure memory map for board peripherals.

Table 5-5: Secure board peripherals

Row ID	Address range in secure region		Size	Description	Modeled in FVP
	From	To			
1	0x5110_0000	0x5110_0FFF	4KB	GPIO 0	CMSDK GPIO
2	0x5110_1000	0x5110_1FFF	4KB	GPIO 1	CMSDK GPIO
3	0x5110_2000	0x5110_2FFF	4KB	GPIO 2	CMSDK GPIO
4	0x5110_3000	0x5110_3FFF	4KB	GPIO 3	CMSDK GPIO
5	0x5110_4000	0x5110_4FFF	4KB	USER AHB 0	Not modeled
6	0x5110_5000	0x5110_5FFF	4KB	USER AHB 1	Not modeled
7	0x5110_6000	0x5110_6FFF	4KB	USER AHB 2	Not modeled
8	0x5110_7000	0x5110_7FFF	4KB	USER AHB 3	Not modeled
-	0x5110_8000	0x513F_FFFF	-	Reserved	-
9	0x5140_0000	0x514F_FFFF	1MB	Ethernet	SMSC91C111 Ethernet controller
10	0x5150_0000	0x515F_FFFF	1MB	USB	Dummy stub
-	0x5160_0000	0x516F_FFFF	-	Reserved	-
11	0x5170_0000	0x5170_0FFF	4KB	Timing Adapter APB0 - FPGA SRAM	Dummy stub
12	0x5170_1000	0x5170_1FFF	4KB	Timing Adapter APB1 - QSPI	Dummy stub
13	0x5170_2000	0x5170_2FFF	4KB	Timing Adapter APB2 - DDR4	Dummy stub
14	0x5170_3000	0x5170_3FFF	4KB	User APB3	Not modeled
-	0x5170_4000	0x517F_FFFF	-	Reserved	-
15	0x5180_0000	0x5180_0FFF	4KB	QSPI Config	Not modeled
16	0x5180_1000	0x5180_1FFF	4KB	QSPI Write	Not modeled
-	0x5180_2000	0x56FF_FFFF	-	Reserved	-
17	0x5700_0000	0x5700_0FFF	4KB	SRAM Memory Protection Controller (MPC)	Modeled
18	0x5700_1000	0x5700_1FFF	4KB	QSPI Memory Protection Controller (MPC)	Modeled
19	0x5700_2000	0x5700_2FFF	4KB	DDR4 Memory Protection Controller (MPC)	Modeled
-	0x5700_3000	0x57FF_FFFF	-	Reserved	-
-	0x5800_0000	0x5810_1FFF	-	Subsystem peripherals	-
-	0x5810_2000	0x591F_FFFF	-	Reserved	-

Row ID	Address range in secure region		Size	Description	Modeled in FVP
	From	To			
20	0x5920_0000	0x5920_0FFF	4KB	FPGA - SBCon I2C (Touch)	Partially modeled
21	0x5920_1000	0x5920_1FFF	4KB	FPGA - SBCon I2C (Audio Conf)	Dummy stub
22	0x5920_2000	0x5920_2FFF	4KB	FPGA - PL022 (SPI ADC)	Dummy stub
23	0x5920_3000	0x5920_3FFF	4KB	FPGA - PL022 (SPI Shield0)	Dummy stub
24	0x5920_4000	0x5920_4FFF	4KB	FPGA - PL022 (SPI Shield1)	Dummy stub
25	0x5920_5000	0x5920_5FFF	4KB	SBCon (I2C - Shield0)	Dummy stub
26	0x5920_6000	0x5920_6FFF	4KB	SBCon (I2C - Shield1)	Dummy stub
27	0x5920_7000	0x5920_7FFF	4KB	USER APB	Dummy stub
28	0x5920_8000	0x5920_8FFF	4KB	DDR4 EEPROM	Dummy stub
-	0x5920_9000	0x592F_FFFF	-	Reserved	-
29	0x5930_0000	0x5930_0FFF	4KB	FPGA - SCC registers	Modeled
30	0x5930_1000	0x5930_1FFF	4KB	FPGA - I2S (Audio)	Partially modeled
31	0x5930_2000	0x5930_2FFF	4KB	FPGA - IO (System Ctrl + I/O)	Modeled
32	0x5930_3000	0x5930_3FFF	4KB	UART0 - UART_F [0]	CMSDK UART
33	0x5930_4000	0x5930_4FFF	4KB	UART1 - UART_F [1]	CMSDK UART
34	0x5930_5000	0x5930_5FFF	4KB	UART2 - UART_F [2]	CMSDK UART
35	0x5930_6000	0x5930_6FFF	4KB	UART3 - UART Shield 0	Dummy stub
36	0x5930_7000	0x5930_7FFF	4KB	UART4 - UART Shield 1	Dummy stub
37	0x5930_8000	0x5930_8FFF	4KB	UART5 - UART_F [3]	CMSDK UART
-	0x5930_9000	0x5930_9FFF	4KB	Reserved	-
38	0x5930_A000	0x5930_AFFF	4KB	CLCD Config Reg	Partially modeled
39	0x5930_B000	0x5930_BFFF	4KB	RTC	PL031_RTC

5.2.5 Corstone SSE-310 FVP expansion peripheral interrupt map

Interrupt map at the board layer.

Table 5-6: Interrupt map at the board layer

Interrupt input	Interrupt source
IRQ [32]	System timestamp counter interrupt. Not implemented.
IRQ [33]	UART 0 receive interrupt.
IRQ [34]	UART 0 transmit interrupt.

Interrupt input	Interrupt source
IRQ [35]	UART 1 receive interrupt.
IRQ [36]	UART 1 transmit interrupt.
IRQ [37]	UART 2 receive interrupt.
IRQ [38]	UART 2 transmit interrupt.
IRQ [39]	UART 3 receive interrupt.
IRQ [40]	UART 3 transmit interrupt.
IRQ [41]	UART 4 receive interrupt.
IRQ [42]	UART 4 transmit interrupt.
IRQ [43]	UART 0 combined interrupt.
IRQ [44]	UART 1 combined interrupt.
IRQ [45]	UART 2 combined interrupt.
IRQ [46]	UART 3 combined interrupt.
IRQ [47]	UART 4 combined interrupt.
IRQ [48]	UART overflow (0, 1, 2, 3, 4, and 5).
IRQ [49]	Ethernet.
IRQ [50]	Audio I2S.
IRQ [51]	Touch screen.
IRQ [52]	USB.
IRQ [53]	SPI ADC.
IRQ [54]	SPI (shield 0).
IRQ [55]	SPI (shield 1).
IRQ [56]	Reserved.
IRQ [57]	DMA channel 0 interrupt.
IRQ [58]	DMA channel 1 interrupt.
IRQ [68:59]	Reserved.
IRQ [69]	GPIO 0 combined interrupt.
IRQ [70]	GPIO 1 combined interrupt.
IRQ [71]	GPIO 2 combined interrupt.
IRQ [72]	GPIO 3 combined interrupt.
IRQ [88:73]	GPIO 0 individual interrupts.
IRQ [104:89]	GPIO 1 individual interrupts.
IRQ [120:105]	GPIO 2 individual interrupts.
IRQ [124:121]	GPIO 3 individual interrupts.
IRQ [125]	UART 5 receive interrupt.
IRQ [126]	UART 5 transmit interrupt.
IRQ [127]	UART 5 combined interrupt.
IRQ [130:128]	Reserved.

5.3 Corstone™ SSE-310 FVP peripheral protection controller expansion map

The Corstone™ SSE-310 FVP implements secure access configuration registers which control security and privileged accesses to peripherals connected to PPC.

Table 5-7: Secure access configuration registers - PPC bits

Bit	MAIN_PPCEXP0 (AHB0)	MAIN_PPCEXP1 (AHB1)	PERIPH_PPCEXP0 (APB0)	PERIPH_PPCEXP1 (APB1)	PERIPH_PPCEXP2 (APB2)
0	GPIO-0	-	Timing adapter APB 0	SBCon I2C (touch screen)	FPGA - SCC registers
1	GPIO-1	-	Timing adapter APB 1	SBCon I2C (audio conf)	FPGA - I2S (audio)
2	GPIO-2	-	Timing adapter APB 2	FPGA PL022 (SPI2 for ADC)	FPGA - GPIO (System Ctrl + I/O)
3	GPIO-3	-	-	FPGA PL022 (SPI Shield0)	UART0
4	User AHB 0	-	-	FPGA PL022 (SPI Shield1)	UART1
5	User AHB 1	-	-	FPGA SBCon (I2C - Shield0)	UART2
6	User AHB 2	-	-	FPGA SBCon (I2C - Shield1)	UART3 STUB
7	User AHB 3	-	-	Reserved	UART4 STUB
8	USB and ethernet	-	-	FPGA - SBCon I2C (DDR4 EPROM)	UART5
9	-	-	-	-	Reserved
10	-	-	-	-	CLCD config reg
11	-	-	-	-	RTC
12	-	-	-	-	-
13	-	-	-	-	-
14	-	-	-	-	-
15	-	-	-	-	-
PPC IRQ no.	5	6	2	3	4

5.4 Corstone™ SSE-310 FVP memory components

The Corstone™ SSE-310 FVP includes the following memory components and their security access is controlled by the MPC.

Table 5-8: Memory components

Name	Non-secure address range	Secure alias	Size
SRAM	0x0100_0000 - 0x011F_FFFF	0x1100_0000 - 0x111F_FFFF	2MB
QSPI SRAM	0x2800_0000 - 0x287F_FFFF	0x3800_0000 - 0x387F_FFFF	8MB

Table 5-9: DDR regions

Name	Address range	IDAU region security	Size
DDR0	0x6000_0000 - 0x6FFF_FFFF	NS	256MB
DDR1	0x7000_0000 - 0x7FFF_FFFF	S	256MB
DDR2	0x8000_0000 - 0x8FFF_FFFF	NS	256MB
DDR3	0x9000_0000 - 0x9FFF_FFFF	S	256MB
DDR4	0xA000_0000 - 0xAFFF_FFFF	NS	256MB
DDR5	0xB000_0000 - 0xBFFF_FFFF	S	256MB
DDR6	0xC000_0000 - 0xCFFF_FFFF	NS	256MB
DDR7	0xD000_0000 - 0xDFFF_FFFF	S	256MB

6. Arm® Corstone™-1000 FVP

This is an example FVP system as specified by the Corstone™-1000 architecture and system specifications.

Refer to the [Arm Corstone-1000 Technical Overview](#) for more information.



You must have installed the MMEncrypt add-on package to use this FVP. It can be downloaded from [Product Download Hub](#).

6.1 Corstone™-1000 FVP modeled components

The following components are present in the Corstone™-1000 FVP.

- Host controller:
 - [Cortex-A35 64-bit 4x Cluster CPU](#)
 - [GIC400](#)
 - [Firewall](#)
- MPS3 board:
 - [SMSC 91C111 ethernet controller](#)
 - [PrimeCell RTC \(PL031\) module](#)
 - Audio I2S and SBCon I2C as dummy stub modules
 - [Two SD cards](#)
 - [DRAM 2GB](#)
 - [SRAM 32MB](#)
 - [PL050 KMI](#)
 - [PS2 keyboard](#)
 - [SP805 watchdog](#)
 - [Virtio-Net](#)
- Secure enclave:
 - [M0+ CPU](#)
 - [CryptoCell312](#)
 - SE Flash (64KB up to 8MB)
 - NVM OTP memory for CC312
 - [Firewall](#)



The FVP does not model Arm® CoreSight™ components.

6.2 Run the Corstone™-1000 FVP with the software package

This FVP is intended to be used with the Arm reference design software package.

About this task

The User Guide for the software package can be found at [ARM Corstone1000](#).

Procedure

1. To see a list of available configuration parameters, run the FVP with the `--list-params` option. Many of these parameters help to bring up bare-metal software on the FVP and are not required if the model is run with the Arm® firmware supplied.
2. This example command line launches the model with the Arm software package:

```
./FVP Corstone-1000 \  
-C diagnostics=4 \  
-C se.trustedBootROMloader.fname="bl1.bin" \  
-C se.trustedSRAM_config=6 \  
-C se.BootROM_config="3" \  
-C se.nvm.raw_image="cc312_otp.bin" \  
--data board.flash0=corstone1000-aarch64-image-mps3.wic.nopt@0x68050000 \  
-C board.xnvm_size=64 \  
-C board.smc_91c111.enabled=1 \  
-C board.hostBridge.userNetworking=true \  
-C board.se_flash_size=8192 \  
-C board.msd_mmc.p_mmc_file=mmc1.dat \  
-C board.msd_mmc.card_type="SD" \  
-C board.msd_mmc.p_fast_access=0 \  
-C board.msd_mmc_2.p_mmc_file=mmc2.dat \  
-C board.msd_mmc_2.card_type="SD" \  
-C board.msd_mmc_2.p_fast_access=0 \  
-C board.msd_mmc.p_max_block_count="0xFFFF" \  
-C board.msd_mmc_2.p_max_block_count="0xFFFF" \  
-C board.flashloader0.fname="flash0" \  
-C board.flashloader0.fnameWrite="flash0" \  
-C board.msd_mmc.support_unpadded_images=true
```


6.3 Corstone™-1000 board peripherals memory and interrupt map

The subsystem model is complemented by a range of peripherals. The board peripherals are representative of peripherals present on the board onto which the SoC is mounted.

Table 6-1: Corstone™-1000 board peripherals memory and interrupt map

Component	Base address	Size	IRQ
Ethernet	0x00_4010_0000	1M	116
SCC registers	0x00_4000_0000	4KB	-
IO registers	0x00_4001_0000	4KB	-
SD card 0	0x00_4030_0000	64KB	149
Virtio-Net	0x00_4040_0000	64KB	145
SD card 1	0x00_5000_0000	64KB	147
SE flash	0x00_6001_0000	8MB	-

6.4 Networking on Corstone™-1000 FVP

Steps to bring up the network on the FVP.

Procedure

1. Set up the TAP interface on the host machine. For information, see [TAP/TUN networking](#) in the *Fast Models Reference Guide*.
2. Append the following parameters to the model:
 - `-C board.smsc_91c111.enabled=true`
 - `-C board.hostbridge.interfaceName="<TAP Interface Name>"`

7. Juno FVP3 model

This model implements the Juno Arm®v8 hardware platform, revision r0.

The Juno SoC is described in the [Juno Arm Development Platform TRM](#).

The version number of the model, including the build number, can be obtained by running the model with the `--version` parameter.

For information about the changes in this release, see the Fast Models Portfolio release notes.

To provide feedback on the product, create a ticket at <https://support.developer.arm.com>.

7.1 Unimplemented features and model limitations

Some peripherals are unimplemented in the model, and others are implemented with limitations.

The model does not implement the following peripherals that are described in the Juno SoC ADP TRM:

- CoreSight™ and device discovery ROMs.
- CoreSight™ devices. The memory area is mapped as a dummy APB region.
- SoC Master signals.
- DFI phy config. It is mapped as a dummy APB region.
- PVT sensors are mapped as DummyAPB peripherals.
- PCIe and the associated SMMU, and the associated MSI mechanism.
- USB and the associated SMMU.
- I2C controllers and their associated peripherals.

The following peripherals are implemented but with limitations:

- The Mali™-T624 GPU is present in a form that permits a driver to load and make forward progress, but it does not render any output to the target surface or buffer.
- 2 x HDLCD controllers are present but the downstream physical ports and logic that would normally be part of selecting a suitable display format are not modeled. An additional Linux driver is available instead.

7.2 Running the Juno FVP3 model

The model can be run through the Arm DS debug IDE or standalone.



When running the model on a Linux host machine, ensure that it has at least 16GB of RAM, in addition to the normal Fast Models requirements. The Juno platform and model incorporate 8GB of DRAM.

7.2.1 Run the Juno FVP3 model in Arm DS

Arm DS can auto-detect and connect to the Juno FVP and offers access to the big.LITTLE™ cores individually or collectively. It is also possible to connect to the Cortex®-M3 core in the SCP.

Start the model with the `--iris-server` option, or the older `--cadi-server` option, then use **New > Model connection** to begin the process of auto-detection.

7.2.2 Run the Juno FVP3 model standalone

The Juno FVP can boot standard software stacks, for example Linux, with minimal modification.

Example command line:

```
$BUILD/Platforms/LISA/CSS/Build_Juno_FVP3/Linux64-Release-GCC-7.3/isis_system \
--plugin $BUILD/PVLIB_HOME/plugins/Linux64_GCC-7.3/Crypto.so \
-C css.aon.scp.ROMloader.fname=juno-scp-rom.bin \
-C board.flashloader0.fname=juno-nor-image.bin \
-C board.base_clk_frequency=0x17D7840 \
-C soc.pl011_uart0.unbuffered_output=1 \
-C soc.pl011_uart1.unbuffered_output=1 \
-C board.pl011_uart1.unbuffered_output=1 \
-C soc.scc.gpr0=0x03000000 \
-C soc.scc.apps_alt_boot=0xBEC0000 \
-C soc.scc.scp_alt_boot=0xABE40000 \
-C board.mmc.p_mmc_file=optional-mmc-card.img \
-C board.virtioblockdevice.image_path=optional-virtio-disk.img \
-C board.smsc_91c111.enabled=1 \
-C board.hostbridge.userNetworking=<0|1> \
-C board.hostbridge.interfaceName=<tap>
```



- `juno-scp-rom.bin` is included with the model in this release.
- `juno-nor-image.bin` represents the contents of the NOR flash memory.

In hardware, there is a complex mechanism to present the NOR flash as a USB disk to allow replacement firmware, for example arm-trusted-firmware, boot loader, or Linux kernel, to be pushed to the board. Instead of modeling this, a script, `images/create_afs_image.py` is included to create the same memory image directly from the firmware components. Any optional MMC image should be less than 2GB.

See [User mode networking](#) for more information about networking in Fast Models.

7.2.3 Run the create_afs_image.py script

This topic shows an example command line that boots Linux using U-Boot.

```
python3 create_afs_image.py juno-nor-image.bin \
${FW_SW_DIR}/fip-uboot.bin,-,0 \
${KERNEL_OUT_DIR}/arch/arm64/boot/Image,Image \
${KERNEL_OUT_DIR}/arch/arm64/boot/dts/arm/juno-fvp.dtb,board.dtb \
${FW_SW_DIR}/scp_b11.bin,-,0x3e40000 \
${FW_SW_DIR}/b11.bin,-,0x3ec0000 \
uboot-script.img,BOOTSCR \
./u-boot-saveenv.bin,-,0x3fc0000
```

The entry for each component of the NOR image consists of:

- The file on the host machine to add to the image.
- An optional alternative name to use for that file when the NOR metadata is generated. Use a '-' to not have an entry in the metadata at all, or leave blank to use the leaf name of the host file.
- An optional address at which to position the file in the NOR image. Those components shown with an address must use that address to place code and data at expected locations. Other components can be referenced by name so their address does not need to be fixed. Components with no fixed address are allocated sequentially after the previous component. There is no intelligent allocation of components to make best use of the available space. A warning occurs if the image exceeds the size of the NOR area.

In this command line:

- `FW_SW_DIR` points to the `SOFTWARE` directory of a standard Juno firmware bundle, for example those from <http://releases.linaro.org/members/arm/platforms/20.01/> and `KERNEL_OUT_DIR` is the build output of a Linux kernel.
- `juno-fvp.dtb` is a modified version of the normal `juno.dts` that removes the unsupported hardware, for example PCIe, USB, and I2C audio, and adds some additional sections, for example the virtio disk and a helper mechanism for the HDLCD output.

`juno-fvp.dts` is included alongside the model in this release and should be used to patch a standard Linux kernel to create the DTB.

- `BOOTSCR/uboot-script.img` allows you to make permanent changes to the boot sequence programmatically. One required change is typically to tell the kernel to boot from the virtio device `/dev/vda` instead of a regular disk at `/dev/sda`. With bigger kernels, it is also necessary to move the device tree up in memory, for example:

```
echo "setenv fdt_addr 83000000" > uboot_script.txt
echo "setenv bootargs \"$bootargs root=/dev/vda" >> uboot_script.txt
mkimage -T script -C none -n 'BOOTSCR' -d uboot-script.txt uboot_script.img
```

If your disk image has no partition table, then `root=/dev/vda` is correct.

If your disk image has a partition table then you need something like `root=/dev/vda2` to point at the correct partition to find the root fs. You can check this with:

```
fdisk -l optional-virtio-disk.img
```

- `u-boot-saveenv.bin` is an optional example of an alternative way to modify U-Boot settings by replacing the region of flash containing the saved U-Boot environment. This is not supplied, but can be created by using the U-Boot commands to modify the environment and save it to another part of memory to be dumped to disk.

7.3 Jump-start the application clusters on bare metal

It is possible to bring the application clusters out of reset without running firmware on the SCP. This is supported using parameters that allow the user to override the default configuration for the SCP control over Cortex®-A53 and Cortex®-A57 cluster reset.

Add the following lines as appropriate to the Juno configuration file:

```
-C css.aon.scp.scp_sc.a53_power_on=15  
-C css.aon.scp.scp_sc.a57_power_on=3
```

The parameter is a bitfield describing which of the CPUs to turn on.



If you are using a separate parameter file, the `-c` is not required. Specify each parameter on a new line in the file.

To supply an image at the same time, add the following:

```
-a css.cluster0.*=<your-image>
```

You can also target `cluster1`, and optionally any supported `cpu` in a cluster, for example `css.cluster0.cpu0`. Because the two clusters and the CPUs within a cluster share a memory interface, this rarely makes a difference.

7.4 Basic setup for the TZC-400 model on bare metal

The default state out of reset for the TZC-400 is to block all accesses to DRAM.

You can configure the TZC-400 through the programmer's view or by using command-line parameters. For many applications, you can ignore TrustZone® security for DRAM. In this case,

you must enable read/write access to DRAM from Secure and Non-Secure world in the application clusters. This can be achieved with the following parameters:

```
-C css.tzc400.rst_gate_keeper=0x0f  
-C css.tzc400.rst_region_attributes_0=0xc000000f  
-C css.tzc400.rst_region_id_access_0=0xffffffff
```

8. Arm® Neoverse™ N1 edge and Neoverse™ E1 edge reference design FVPs

This chapter describes the Arm® Neoverse™ N1 edge and Arm® Neoverse™ E1 edge reference design FVPs. These FVPs are collectively referred to as RD-N1-E1-edge FVPs.

A reference design is a collection of resources, including documentation, a software stack, and FVPs, that describe and model the design choices and performance for recommended configurations of a typical Arm®-based subsystem.

The RD-N1-E1-edge FVPs drive system architecture and software standardization. They provide software and binaries of proprietary firmware that reduce the amount of work that is required for SoC development.



Arm is working to make more content available for these FVPs. To find out more about reference designs, or to contact Arm about them, see [Neoverse Reference Design](#).

8.1 About the RD-N1-E1 FVPs

The RD-N1-E1-edge FVPs model many of the Arm® IP components in the RD-N1-E1-edge design.

The RD-N1-E1-edge FVPs model the following RD-N1-E1-edge configurations:

Config1

N1 2xMP4, 512KB L2 cache per core, 4x2 mesh, 2xDMC.

Config2

E1 2xMP8, 256KB L2 cache per core, 4x2 mesh, 2xDMC.

Config3

Dual-chip. Two Config1 subsystems linked by CMN-600 CML.

The diagrams in [8.2 Block diagrams for RD-N1-E1-edge](#) on page 68 show the IP components that RD-N1-E1-edge describes. Not all of these components are modeled by these FVPs. The following components are modeled:

- N1 2xMP4 or E1 2xMP8 cores.
- System Control Processor (SCP).
- Management Control Processor (MCP).
- CMN-600.

- Multiple NIC-450 interconnects, although NIC-450 is replaced with a simple bus model.
- Memory access path towards DRAM, including DMC-620 memory controllers.

The following components are not modeled:

- CoreSight™ SoC.
- ELA-500.
- USB.

8.2 Block diagrams for RD-N1-E1-edge

The following diagrams show the composition of RD-N1-E1-edge systems.

Figure 8-1: Block diagram for Neoverse N1 edge reference design

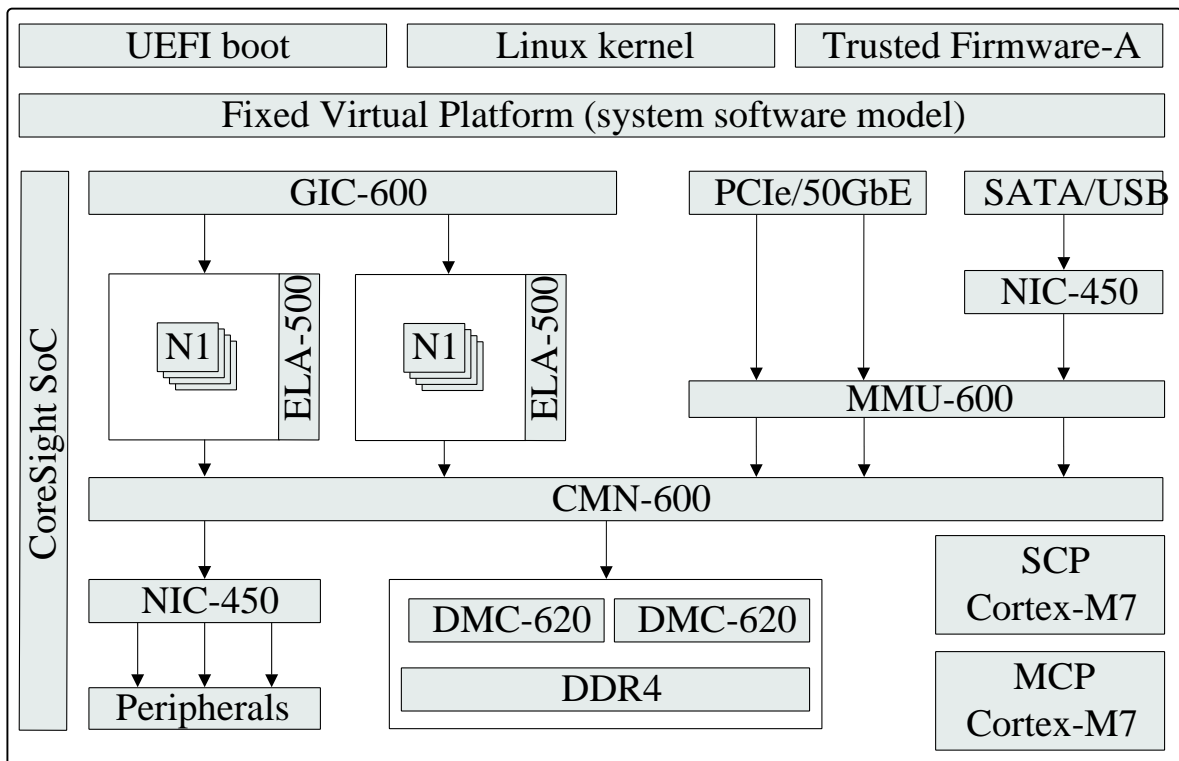
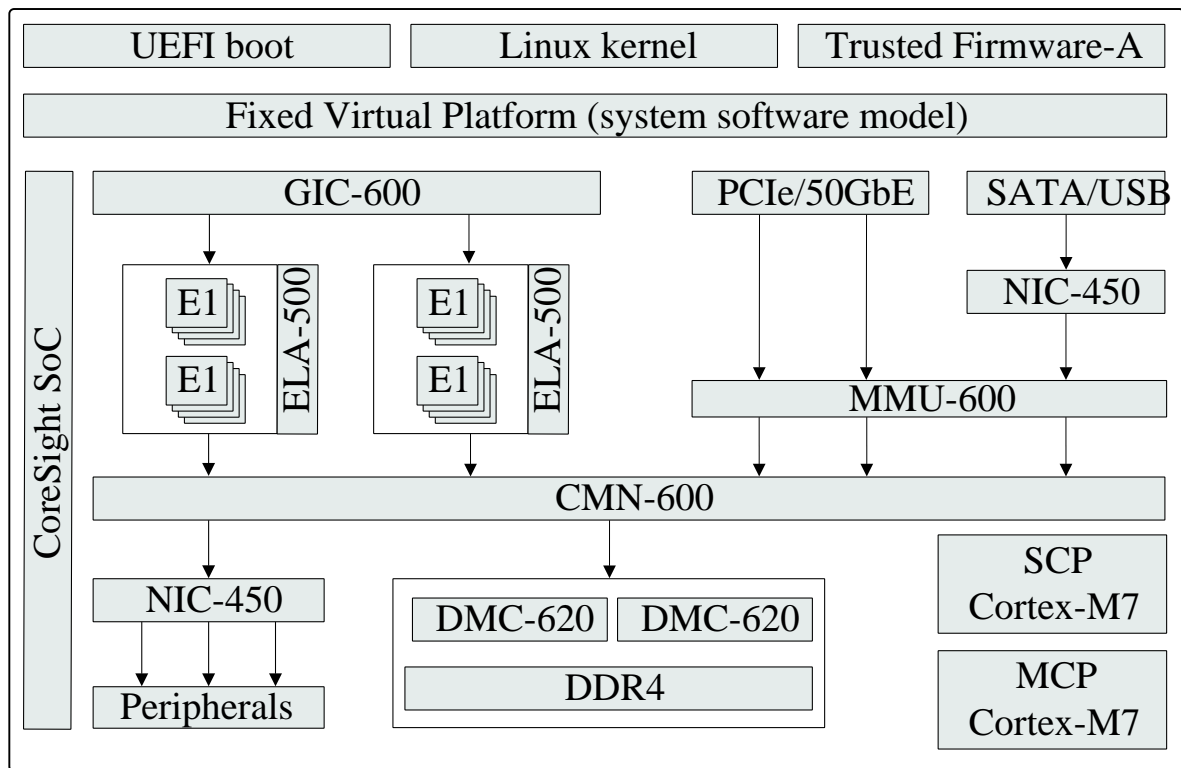


Figure 8-2: Block diagram for Neoverse E1 edge reference design

8.3 RD-N1-E1 FVP peripherals

The RD-N1-E1-edge FVPs include peripherals that the software payload requires to run.

These peripherals are organized in two layers:

SoC

The SoC peripherals represent peripherals that may be added to a compute subsystem in a SoC design.

Board

The board peripherals represent peripherals that may be present on the board onto which the SoC is mounted.

The RD-N1-E1-edge SoC model and board model are based on the Juno Arm® Development Platform (ADP).

In Config3, each compute subsystem has its own SoC and board layers. In this two-chip configuration, each chip is assigned the following memory regions:

Chip 0

0x000_0000_0000-0x3FF_FFFF_FFFF

Chip 1

0x400_0000_0000-0x7FF_FFFF_FFFF

A sideband communication channel is required to coordinate multi-chiplet software boot over CMN-600. The FVP implements this using the MHU device, but Arm recommends using a solution such as I2C in hardware.

8.3.1 Memory map for RD-N1-E1-edge FVP SoC peripherals

This table shows the memory map for the SoC peripherals in the RD-N1-E1-edge FVPs.

**Note**

The SoC peripherals area in the RD_N1_E1_edge memory map is mapped to an Expansion AXI region. Therefore, these mappings are not defined in the reference design.

Table 8-1: SoC peripherals memory map

Name	Base address	Size	Description
SMC interface	0x00_0800_0000	368MB	Routed to board
SMMUv3	0x00_2B40_0000	1MB	-
PCIe config	0x00_6000_0000	16MB	-
PCIe memory	0x00_7000_0000	132MB	-
DMA MMU-400	0x00_7FB0_0000	64KB	-
HDLCD1 MMU-400	0x00_7FB1_0000	64KB	-
HDLCD0 MMU-400	0x00_7FB2_0000	64KB	-
DDR3 PHY 0	0x00_7FB6_0000	64KB	Dummy APB
DDR3 PHY 1	0x00_7FB7_0000	64KB	Dummy APB
DDR3 PHY 2	0x00_7FB8_0000	64KB	Dummy APB
DDR3 PHY 3	0x00_7FB9_0000	64KB	Dummy APB
SoC interconnect NIC-400 GPV	0x00_7FD0_0000	1MB	-
Surge detector	0x00_7FE5_0000	4KB	Dummy APB
TRNG	0x00_7FE6_0000	4KB	-
Trusted non-volatile counters	0x00_7FE7_0000	4KB	-
Trusted Root-Key storage	0x00_7FE8_0000	4KB	-
Secure I2C	0x00_7FE9_0000	256B	A Secure I2C component does not exist in the FVP. Instead, a PL061_GPIO is mapped as a dummy component. It is not functional as a GPIO.
DMA non-secure	0x00_7FF0_0000	4KB	-

Name	Base address	Size	Description
DMA secure	0x00_7FF1_0000	4KB	-
HDLCD1	0x00_7FF5_0000	4KB	-
HDLCD0	0x00_7FF6_0000	4KB	-
UART 1	0x00_7FF7_0000	4KB	-
UART 0	0x00_7FF8_0000	4KB	-
I2S	0x00_7FF9_0000	1KB	An I2S component does not exist in the FVP. Instead, a PL061_GPIO is mapped as a dummy component. It is not functional as a GPIO.
I2C	0x00_7FFA_0000	0x256B	An I2C component does not exist in the FVP. Instead, a PL061_GPIO is mapped as a dummy component. It is not functional as a GPIO.
PL352	0x00_7FFD_0000	4KB	PL354
AP configuration	0x00_7FFE_0000	4KB	GPR
System override registers	0x00_7FFF_0000	4KB	-
PCIe memory	0x05_0000_0000	12GB	-



The following devices are discoverable on the PCIe bus:

- Virtio block device x 2.
- AHCI controller with attached SATA disk.

8.3.2 Memory map for RD-N1-E1-edge FVP board peripherals

This table shows the memory map for the board peripherals in the RD-N1-E1-edge FVPs.



The board peripherals area in the RD_N1_E1_edge memory map is mapped to an Expansion AXI region. Therefore, these mappings are not defined in the reference design.

Table 8-2: Board peripherals memory map

Name	Base address	Size	Description
NOR Flash 0	0x00_0800_0000	64MB	-
NOR Flash 1	0x00_0C00_0000	64MB	-
NOR Flash 2	0x00_1000_0000	64MB	-
Ethernet	0x00_1800_0000	64MB	SMSC 91C111
System registers	0x00_1C01_0000	64KB	-
MCI	0x00_1C05_0000	64KB	PL180
KMI 0	0x00_1C06_0000	64KB	PL050
KMI 1	0x00_1C07_0000	64KB	PL050
UART 0	0x00_1C09_0000	64KB	PL011

Name	Base address	Size	Description
UART 1	0x00_1C0A_0000	64KB	PL011
Watchdog	0x00_1C0F_0000	64KB	SP805
Dual timer	0x00_1C11_0000	64KB	SP804
Virtio block device	0x00_1C13_0000	64KB	-
Virtio net device	0x00_1C15_0000	64KB	-
RTC	0x00_1C17_0000	64KB	PL031
GPIO 0	0x00_1C1D_0000	64KB	PL061_GPIO is mapped as a dummy component in the FVP. It is not functional as a GPIO.
GPIO 1	0x00_1C1E_0000	64KB	PL061_GPIO is mapped as a dummy component in the FVP. It is not functional as a GPIO.
DRAM	0x00_8000_0000	2GB	-
DRAM	0x80_8000_0000	6GB	-

8.3.3 Interrupt maps for RD-N1-E1-edge FVP

These tables show the interrupt IDs and sources for the FVP peripherals.

Table 8-3: Interrupt map at the SoC layer

Interrupt ID	Source	Description
147	UART 0	-
148	UART 1	-
171	TRNG	-

Table 8-4: Interrupt map at the board layer

Interrupt ID	Source	Description
111	Ethernet	-
132	RTC	-
133	UART 0	-
134	UART 1	-
140	VFS2	-
202	Virtio	-
204	Virtio net device	-
228	Watchdog	-
229	KMI 0	-
230	Dual timer	Interrupts 0 and 1
231	System registers ethernet IRQ	-

9. Arm® Neoverse™ N2 reference design FVP

The Arm® Neoverse™ N2 reference design (RD-N2) Fixed Virtual Platform (FVP) models much of the Arm® IP in RD-N2. The FVP enables partners to develop ahead of hardware availability and to explore the design from a software perspective.

The FVP is used with the RD-N2 software package. For instructions on setting up and running the FVP, see the RD-N2 Software Bundle Readme.

9.1 About the RD-N2 FVP

The RD-N2 FVP models CFG32C4M, which is a single-chiplet system with the number of Arm® Neoverse™ N2 cores reduced to 16.

The FVP models the following IP components:

- Arm® Neoverse™ N2 MP1 cores.
- CMN-700.



The CMN-700 model supports RAS, which when enabled can impact system performance. To avoid this, the model disables RAS by default. To enable it, set the CMN700 model parameter `enable_ras` to true.

-
- Multiple NIC-450 interconnects.
 - GIC-700.
 - MMU-700.
 - Arm® Cortex®-M7 SCP and MCP cores.



The FVP does not model every component that RD-N2 describes. For example, the FVP does not model the CoreSight™ technology components.

The RD-N2 FVP drives system architecture and software standardization. The models provide software and binaries of proprietary firmware that reduce the amount of work that is required for SoC development.

9.2 RD-N2 FVP peripherals

The RD-N2 FVP includes peripherals that the software payload requires to run.

The SoC peripherals area and board peripherals area in the RD-N2 memory map are mapped to an expansion AMBA® AXI region. Therefore, these mappings are not defined in the reference design.

The peripherals are organized into two layers:

SoC

The SoC peripherals represent peripherals that might be added to a compute subsystem in an SoC design. The RD-N2 SoC model is based on the Juno Arm® Development Platform (ADP).

Board

The board peripherals represent peripherals that might be present on the board onto which the SoC is mounted. The RD-N2 board model is based on the Juno ADP.

9.2.1 Memory map for RD-N2 FVP SoC peripherals

This table shows the memory map for the SoC peripherals in the RD-N2 FVP.

Table 9-1: RD-N2 FVP SoC peripherals

Name	Base address	Size	Description
SMC0 interface	0x00_0800_0000	64MB	Routed to the board.
SMC1 interface	0x10_5000_0000	256MB	Routed to the board.
PCIe config	0x10_1000_0000	256MB	-
PCIe memory	0x00_6000_0000	512MB	-
DMA MMU-500	0x00_E00_0000	64KB	-
HDLCD1 MMU-500	0x00_E01_0000	64KB	-
HDLCD0 MMU-500	0x00_E02_0000	64KB	-
SoC interconnect NIC-400 GPV	0x00_ED0_0000	1MB	-
Surge detector	0x00_EE5_0000	4KB	Dummy APB.
TRNG	0x00_EE6_0000	4KB	-
Trusted non-volatile counters	0x00_EE7_0000	4KB	-
Trusted root-key storage	0x00_EE8_0000	4KB	-
Secure I2C	0x00_EE9_0000	512 bytes	There is no Secure I2C component in the FVP. Instead, a PL061 GPIO is mapped as a dummy component.
DDR PHY 0-31	0x00_EB0_0000	2MB	Dummy APB.
DMA secure	0x00_EF0_0000	64KB	-
DMA non-secure	0x00_EF1_0000	64KB	-
PCIe macro	0x00_EF2_0000	64KB	-
PCIe root port	0x00_EF3_0000	64KB	-

Name	Base address	Size	Description
HDLCD1	0x00_EF5_0000	4KB	-
HDLCD0	0x00_EF6_0000	4KB	-
UART 0	0x00_EF7_0000	4KB	-
UART 1	0x00_EF8_0000	4KB	-
I2S	0x00_EF9_0000	1KB	There is no I2S component in the FVP. Instead, a PL061 GPIO is mapped as a dummy component.
I2C	0x00_EFA_0000	256 bytes	There is no I2C component in the FVP. Instead, a PL061 GPIO is mapped as a dummy component.
PL354	0x00_EFD_0000	64KB	Arm® PrimeCell SMC dual SRAM memory interface (PL354).
System override registers	0x00_EFF_0000	64KB	-
AP configuration	0x00_EFE_0000	64KB	Granular Power Requester (GPR).

9.2.2 Memory map for RD-N2 FVP board peripherals

This table shows the memory map for the board peripherals in the RD-N2 FVP.

Table 9-2: RD-N2 FVP board peripherals

Name	Base address	Size	Description
NOR flash 0	0x00_0800_0000	64MB	-
NOR flash 1	0x10_5000_0000	64MB	-
NOR flash 2	0x10_5400_0000	64MB	-
Ethernet	0x10_5C00_0000	64MB	Non-PCI Ethernet controller (SMSC 91C111)
System registers	0x00_0C01_0000	64KB	-
SP810 Sysctrl	0x00_0C02_0000	64KB	-
MCI	0x00_0C05_0000	64KB	Arm® PrimeCell Multimedia Card Interface (PL180)
KMI 0	0x00_0C06_0000	64KB	Arm® PrimeCell PS2 Keyboard/Mouse Interface (PL050)
KMI 1	0x00_0C07_0000	64KB	Arm® PrimeCell PS2 Keyboard/Mouse Interface (PL050)
UART 0	0x00_0C09_0000	64KB	Arm® PrimeCell UART (PL011)
UART 1	0x00_0C0A_0000	64KB	Arm® PrimeCell UART (PL011)
Watchdog	0x00_0C0F_0000	64KB	Arm® Watchdog Module (SP805)
Dual timer	0x00_0C11_0000	64KB	Arm® Dual-Timer Module (SP804)
Virtio block device	0x00_0C13_0000	64KB	-
Virtio net device	0x00_0C15_0000	64KB	-
GPIO 2Wire (DVI)	0x00_0C16_0000	64KB	Arm® PrimeCell General Purpose Input/Output (PL061)
RTC0	0x00_0C17_0000	64KB	Arm® PrimeCell Real Time Clock (PL031)
RTC1	0x00_0C18_0000	64KB	Arm® PrimeCell Real Time Clock (PL031)
GPIO 0	0x00_0C1D_0000	64KB	Arm® PrimeCell General Purpose Input/Output (PL061)
GPIO 1	0x00_0C1E_0000	64KB	Arm® PrimeCell General Purpose Input/Output (PL061)
DRAM	0x00_8000_0000	2GB	-

Name	Base address	Size	Description
DRAM	0x80_8000_0000	6GB	-

9.2.3 Interrupt maps for RD-N2 FVP

These tables show the interrupt IDs and sources for the FVP peripherals.

Table 9-3: Interrupt map at the SoC layer

Interrupt ID	Source
403	UART 0
404	UART 1
427	TRNG

Table 9-4: Interrupt map at the board layer

Interrupt ID	Source
387	RTC1
388	RTC0
389	UART0 (board)
390	UART1 (board)
391	KMI1
392	GPIO0
393	GPIO1
394	I2C GPIO
395	MCIINTR0
396	MCIINTR1
397	SMSC 91C111
405	HDLCD controller 0
406	SMC PL354 interface 0
407	SMC PL354 interface 1
413	HDLCD controller 1
414	SMMU combined Secure interrupt
415	SMMU combined Non-secure interrupt
418-425	DMA0 IRQ7-0
426	DMA0 IRQ abort
428-435	DMA1 IRQ7-0
436	DMA1 IRQ abort
458	Virtio block device
460	Virtio net
483	RTCC
484	WDT

Interrupt ID	Source
485	KMIO
486	Dual timer
487	System registers
488	System register – USB
489	System register – Tile
490	System register – Push button
491	System register – Ethernet

10. Arm® Neoverse™ V1 reference design FVP

To develop ahead of hardware availability and to explore the design from a software perspective, the Fixed Virtual Platform (FVP) models many of the Arm® IP in the RD-V1 design.

10.1 About the RD-V1 FVP

Two configurations of RD-V1 are supported.

- RD-V1 FVP models Config-M, a single-chiplet system with 16 Arm® Neoverse™ V1 cores.
- RD-V1 quad-chiplet FVP models a reduced-size variant of Config-XL, consisting of four compute subsystems linked by CMN-650 CML. It provides a functional model of a quad-chiplet system. Each subsystem contains four Arm® Neoverse™ V1 cores, for a total of 16 cores in the FVP (at full size, Config-XL has 4 x 32 cores).

The FVP models the following IP components:

- Arm® Neoverse™ V1 MP1.
- GIC-700.
- MMU-600.
- SCP.
- MCP.
- CMN-650.
- Multiple NIC-450 interconnects.
- Memory access path towards DRAM which includes a TrustZone® controller.



The FVP does not model every component that RD-V1 describes. For example, it does not model the CoreSight™ technology components.

The RD-V1 FVP drives system architecture and software standardization. The models provide software and binaries of proprietary firmware that reduce the amount of work that is required for SoC development.

10.2 RD-V1 FVP peripherals

The RD-V1 FVP includes peripherals that the software payload requires to run.

These peripherals are organized in two layers:

SoC

The SoC peripherals represent peripherals that can be added to a compute subsystem in a SoC design. The Arm® Neoverse™ V1 reference design SoC model is based on the Juno Arm Development Platform (ADP).

Board

The board peripherals represent peripherals that can be present on the board onto which the SoC is mounted. The RD-V1 board model is based on the Juno Arm Development Platform (ADP).

In the quad-chiplet system, each compute subsystem has SoC and Board layers dedicated to that subsystem. Addressing for each chip is defined in chapter 7.1, AP memory map, in the Arm® Neoverse™ V1 reference design Software Developer Guide.

Chip 0	0x000_0000_0000-0x3FF_FFFF_FFFF
Chip 1	0x400_0000_0000-0x7FF_FFFF_FFFF
Chip 2	0x800_0000_0000-0xBFF_FFFF_FFFF
Chip 3	0xC00_0000_0000-0xFFF_FFFF_FFFF

A sideband communication channel is required to coordinate multi-chiplet software boot over CMN-650. The FVP implements this using the MHU device, but Arm recommends using a solution such as I²C in hardware.

Related information

[Arm Neoverse v1 reference design Software Developer Guide](#)

10.2.1 Memory map for RD-V1 FVP SoC peripherals

This table shows the memory map for the SoC peripherals in the RD-V1 FVP.



Note

The following devices are discoverable on the PCIe bus:

- Virtio block device x 2.
- AHCI controller with attached SATA disk.

Table 10-1: SoC peripherals

Name	Base address	Size	Description
SMC interface	0x00_0800_0000	368MB	Routed to Board
SMMUv3	0x00_2B40_0000	1MB	-
PCIe Config	0x00_6000_0000	16MB	-

Name	Base address	Size	Description
PCIe Memory	0x00_7000_0000	132MB	-
DMA MMU-400	0x00_7FB0_0000	64KB	-
HDLCD1 MMU-400	0x00_7FB1_0000	64KB	-
HDLCD0 MMU-400	0x00_7FB2_0000	64KB	-
DDR3 PHY 0	0x00_7FB6_0000	64KB	Dummy APB
DDR3 PHY 1	0x00_7FB7_0000	64KB	Dummy APB
DDR3 PHY 2	0x00_7FB8_0000	64KB	Dummy APB
DDR3 PHY 3	0x00_7FB9_0000	64KB	Dummy APB
SoC Interconnect NIC-400 GPV	0x00_7FD0_0000	1MB	-
Surge detector	0x00_7FE5_0000	4KB	Dummy APB
TRNG	0x00_7FE6_0000	4KB	-
Trusted Non-volatile counters	0x00_7FE7_0000	4KB	-
Trusted Root-Key storage	0x00_7FE8_0000	4KB	-
Secure I2C	0x00_7FE9_0000	256B	A Secure I2C component does not exist in the FVP. Instead, a PL061_GPIO is mapped.
DMA Non-secure	0x00_7FF0_0000	4KB	-
DMA Secure	0x00_7FF1_0000	4KB	-
HDLCD1	0x00_7FF5_0000	4KB	-
HDLCD0	0x00_7FF6_0000	4KB	-
UART 1	0x00_7FF7_0000	4KB	-
UART 0	0x00_7FF8_0000	4KB	-
I2S	0x00_7FF9_0000	1KB	An I2S component does not exist in the FVP. Instead, a PL061_GPIO is mapped.
I2C	0x00_7FFA_0000	256B	An I2C component does not exist in the FVP. Instead, a PL061_GPIO is mapped.
PL352	0x00_7FFD_0000	4KB	PL354
AP configuration	0x00_7FFE_0000	4KB	GPR
System override Registers	0x00_7FFF_0000	4KB	-
PCIe Memory	0x05_0000_0000	12GB	-

10.2.2 Memory map for RD-V1 FVP board peripherals

This table shows the memory map for the board peripherals in the RD-V1 FVP.



The SoC peripherals area and board peripherals area in the RD-V1 memory map are mapped to an Expansion AXI region. Therefore, these mappings are not defined in the reference design.

Table 10-2: Board peripherals

Name	Base address	Size	Description
NOR Flash 0	0x00_0800_0000	64MB	-
NOR Flash 1	0x00_0C00_0000	64MB	-
NOR Flash 2	0x00_1000_0000	64MB	-
Ethernet	0x00_1800_0000	64MB	SMSC 91C111
System registers	0x00_1C01_0000	64KB	-
MCI	0x00_1C05_0000	64KB	PL180
KMI 0	0x00_1C06_0000	64KB	PL050
KMI 1	0x00_1C07_0000	64KB	PL050
UART 0	0x00_1C09_0000	64KB	PL011
UART 1	0x00_1C0A_0000	64KB	PL011
Watchdog	0x00_1C0F_0000	64KB	SP805
Dual Timer	0x00_1C11_0000	64KB	SP804
Virtio Block Device	0x00_1C13_0000	64KB	-
Virtio Net Device	0x00_1C15_0000	64KB	-
RTC	0x00_1C17_0000	64KB	PL031
GPIO 0	0x00_1C1D_0000	64KB	PL061_GPIO
GPIO 1	0x00_1C1E_0000	64KB	PL061_GPIO
DRAM	0x00_8000_0000	2GB	-
DRAM	0x80_8000_0000	6GB	-

10.2.3 Interrupt maps for RD-V1 FVP

These tables show the interrupt IDs and sources for the FVP peripherals.

Table 10-3: Interrupt map at the SoC layer

Interrupt ID	Source	Description
147	UART 0	-
148	UART 1	-
171	TRNG	-

Table 10-4: Interrupt map at the board layer

Interrupt ID	Source	Description
111	Ethernet	-
132	RTC	-
133	UART 0	-
134	UART 1	-
140	VFS2	-
202	Virtio	-

Interrupt ID	Source	Description
204	Virtio net device	-
228	Watchdog	-
229	KMI 0	-
230	Dual Timer	Interrupts 0 and 1
231	System registers Ethernet IRQ	-

11. Configurable PCIe hierarchy

Configurable PCIe hierarchy allows you to create a customized PCIe hierarchy based on a JSON-formatted file.

To use it, add the following parameter to the command line before running the model, specifying the path to the PCIe hierarchy file:

```
-C pci.hierarchy_file_name=/path/to/hierarchy.json
```



PCIe hierarchy files are supported by most Infrastructure Reference Design FVPs. To find out if your platform supports it, run it with the `--list-params` option to see if it has the `*.hierarchy_file_name` parameter.

11.1 JSON format for the hierarchy

To represent an endpoint device, a root port, or a switch, use the following JSON format.

An endpoint device is represented in the following format:

```
{
  "<device_type>/<device_name>": {
    "parameter1":<parameter_value>
    "parameter2":<parameter_value>
    "  ...  " :<  ...  >
  }
}
```

A root port is represented in the following format, where a `__downstream__` parameter denotes a device connected downstream of the port:

```
{
  "rootport/<rootport_name>": {
    "parameter1":<parameter value>
    "parameter2":<parameter value>
    "  ...  " :<  ...  >
    "__downstream__":{
      "<device_type>/<device_name>":{
        "parameter3":<parameter value>
        "parameter4":<parameter value>
        "  ...  " :<  ...  >
      }
    }
  }
}
```

A switch is represented in the following format, where a `__downstream__<device_number>` parameter denotes the device connected downstream of the switch, with the downstream port at device number `<device_number>` on the switch's internal bus:

```
{
  "switch/<switch_name>": {
    "parameter1":<parameter value>
    "parameter2":<parameter value>
    " ... ":< ... >
    "__downstream__<device_number1>": {
      "<device_type>/<device_name>":{
        "parameter3":<parameter value>
        "parameter4":<parameter value>
        " ... ":< ... >
      }
    },
    "__downstream__<device_number2>":{
      "<device_type>/<device_name>":{
        "parameter5":<parameter value>
        "parameter6":<parameter value>
        " ... ":< ... >
      }
    }
  }
}
```

11.2 Common PCIe endpoint parameters

A PCIe endpoint is the common PCIe frontend for all endpoint-type devices in the hierarchy, namely AHCI controller, host bridge, and SMMU test engine, which are described later in this chapter.

The following table describes the parameters for PCIe-related features of endpoints:

Table 11-1: Common endpoint parameters

Parameter	Description	Type	Range
device	Device number of the endpoint.	int	0-31
function	Function number for the endpoint.	int	0-7
vendor_id	PCI vendor ID.	int	0-0xFFFFE
device_id	PCI device ID.	int	0-0xFFFFE
base_class	PCI base class.	int	0-255
sub_class	PCI sub class.	int	0-255
bar0_64bit	If BAR 0 is 64 bits wide, if region size is nonzero.	bool	-
bar1_64bit	If BAR 1 is 64 bits wide, if region size is nonzero.	bool	-
bar2_64bit	If BAR 2 is 64 bits wide, if region size is nonzero.	bool	-
bar3_64bit	If BAR 3 is 64 bits wide, if region size is nonzero.	bool	-
bar4_64bit	If BAR 4 is 64 bits wide, if region size is nonzero.	bool	-

Parameter	Description	Type	Range
bar0_log2_size	Log2 of the size of the region pointed to by BAR 0. Zero is reserved which means bar is not used ⁷ .	int	0-63
bar1_log2_size	Log2 of the size of the region pointed to by BAR 1. Zero is reserved which means bar is not used.	int	0-63
bar2_log2_size	Log2 of the size of the region pointed to by BAR 2. Zero is reserved which means bar is not used.	int	0-63
bar3_log2_size	Log2 of the size of the region pointed to by BAR 3. Zero is reserved which means bar is not used.	int	0-63
bar4_log2_size	Log2 of the size of the region pointed to by BAR 4. Zero is reserved which means bar is not used.	int	0-63
bar5_log2_size	Log2 of the size of the region pointed to by BAR 5. Zero is reserved which means bar is not used.	int	0-63
uses_interrupt	Enable support for legacy interrupts.	bool	-
interrupt_pin_index	Interrupt pin used by this function. 1 is INTA, 2 is INTB, 3 is INTC, 4 is INTD.	int	1-4
multi_function	Set to true if this endpoint is part of a multi-function device.	bool	-
pcie_version	PCIe version, bits[3:0] in capabilities register. 1 is PCIe 3.0. 2 is PCIe 4.0.	int	1-15
prog_iface	PCI programming interface.	int	0-15
rev_id	PCI revision ID.	int	0-15
subsys_id	PCI subsystem ID.	int	0-0xFFFFE
subsys_vendor_id	PCI subsystem vendor ID.	int	0-0xFFFFE
express_capability_device_type	PCI Express Capabilities Device Type, bits[7:4] in capabilities register. 0 is PCIe EndPoint, 9 is RCiEP.	int	0-15
msix_support	Enable device support for MSI-X.	bool	-
msix_pba_bar	BAR used by MSI-X pending bit array.	int	0-5
msix_table_bar	BAR used by MSI-X table.	int	0-5
msix_table_size	Size of tables for MSI-X.	int	0-2048
power_mgmt_capability	Device supports power-management capabilities.	bool	-
pasid_supported	If set, then the PCIe device can emit PASID (SubstreamIDs).	bool	-
ext_fmt_field_supported	Enable extended format field support.	bool	-
extended_tag_supported	Extended tag field support.	bool	-
link_port_number	Port number for PCIe link.	int	0-255
pri_supported	If set, then the PCIe function supports Page Request Interface (requires ATS).	bool	-
rber_supported	Enable role-based error reporting.	bool	-
slot_and_root_status_msi_idx	MSI index for reporting slot and root status changes.	int	0-2047
tag_10bit_completer_supported	Enable 10-bit tag completer support.	bool	-
tag_10bit_requester_supported	Enable 10-bit tag requester support.	bool	-
aspm_optionality_compliant	Enable ASPM optionality compliance.	bool	-

⁷ Parameter value is fixed and cannot be overridden.

Parameter	Description	Type	Range
ats_supported	If set, then the PCIe function supports Address Translation Services (ATS).	bool	-

11.3 AHCI controller parameters

The Advanced Host Controller Interface (AHCI), is a technical standard for an interface that enables software to communicate with Serial ATA (SATA) devices.

Default values for endpoint parameters:

- device: 0
- function: 0
- vendor_id: 0xABC*
- device_id: 0xACED*
- base_class: 0x1* (device class for mass storage)
- sub_class: 0x6* (sub class for SATA)
- bar0_64bit: false
- bar1_64bit: false
- bar2_64bit: false
- bar3_64bit: false
- bar4_64bit: false
- bar0_log2_size: 13
- bar1_log2_size: 13
- bar2_log2_size: 12
- bar3_log2_size: 13
- bar4_log2_size: 12
- bar5_log2_size: 13
- uses_interrupt: false*
- interrupt_pin_index: 1
- multi_function: false
- pcie_version: 2
- prog_iface: 0x1*
- rev_id: 0x1*
- subsys_id: 0x2*
- subsys_vendor_id: 0x13B5*

- `express_capability_device_type`: 0
- `msix_support`: true*
- `msix_pba_bar`: 4*
- `msix_table_bar`: 2*
- `msix_table_size`: 1*
- `power_mgmt_capability`: true
- `pasid_supported`: false
- `ext_fmt_field_supported`: true
- `extended_tag_supported`: true
- `link_port_number`: 0
- `pri_supported`: false
- `rber_supported`: true
- `slot_and_root_status_msi_idx`: 0
- `tag_10bit_completer_supported`: true
- `tag_10bit_requester_supported`: true
- `aspm_optionality_compliant`: true
- `ats_supported`: false



* indicates parameter values that are fixed and cannot be overridden.

Table 11-2: AHCI parameters and their default values

Parameter	Description	Default value
<code>force_mode</code>	Force disk to report support for at most PIO/DMA/NCQ mode (only for testing/bring-up purposes). PIO mode is always supported. Use NCQ for maximum performance.	"NCQ"
<code>image_path</code>	Comma-separated list of 0-32 disk images. Each image represents one SATA disk which is connected to one port of the AHCI controller. Empty list elements are allowed and result in a SATA port that has no disk attached. An empty string (default) means one SATA port with no disk attached.	""
<code>run_async</code>	Do host I/O in a background thread asynchronously. Enabling this parameter makes the simulation non-deterministic and may or may not improve performance.	false

Example

```
"ahci/ahci0": {
  "device": 10,
  "function": 0,
  "image_path": "/path/to/image",
  "express_capability_device_type": 9
}
```

11.4 Host bridge parameters

On real systems, the host bridge connects the tree of PCI buses, which are internally connected with PCI-to-PCI bridges, to the rest of the system. On FVPs, it is merely a placeholder representation, and does not provide any functionality.

Default values for endpoint parameters:

- `device`: 0
- `function`: 0
- `vendor_id`: 0x13B5
- `device_id`: 0
- `base_class`: 0x6*
- `sub_class`: 0
- `bar0_64bit`: false
- `bar1_64bit`: false
- `bar2_64bit`: false
- `bar3_64bit`: false
- `bar4_64bit`: false
- `bar0_log2_size`: 12
- `bar1_log2_size`: 0
- `bar2_log2_size`: 0
- `bar3_log2_size`: 0
- `bar4_log2_size`: 0
- `bar5_log2_size`: 0
- `uses_interrupt`: false
- `interrupt_pin_index`: 1
- `multi_function`: false
- `pcie_version`: 2
- `prog_iface`: 0xf
- `rev_id`: 0xf
- `subsys_id`: 0xf
- `subsys_vendor_id`: 0x13B5
- `express_capability_device_type`: 9
- `msix_support`: false
- `msix_pba_bar`: 6

- `msix_table_bar`: 6
- `msix_table_size`: 1
- `power_mgmt_capability`: true
- `pasid_supported`: false
- `ext_fmt_field_supported`: true
- `extended_tag_supported`: true
- `link_port_number`: 0
- `pri_supported`: true
- `rber_supported`: true
- `slot_and_root_status_msi_idx`: 0
- `tag_10bit_completer_supported`: true
- `tag_10bit_requester_supported`: true
- `aspm_optionality_compliant`: true



* indicates parameter values that are fixed and cannot be overridden.

Example

```
"hostbridge/hb": {  
    "device": 0,  
    "function": 0,  
}
```

11.5 SMMU test engine parameters

SMMUV3TestEngine is a PCIe device used to generate various stimuli that can help test SMMU integration in the system.

Default values for endpoint parameters:

- `device`: 0
- `function`: 0
- `vendor_id`: 0x13B5*
- `device_id`: 0xFF80
- `base_class`: 0xFF*
- `sub_class`: 0
- `bar0_64bit`: true

- bar1_64bit: false
- bar2_64bit: true
- bar3_64bit: false
- bar4_64bit: true
- bar0_log2_size: 18
- bar1_log2_size: 0
- bar2_log2_size: 15*
- bar3_log2_size: 0
- bar4_log2_size: 12*
- bar5_log2_size: 0
- uses_interrupt: false
- multi_function: false
- pcie_version: 2
- prog_iface: 0x0
- rev_id: 0xf
- subsys_id: 0x0
- subsys_vendor_id: 0x13B5*
- express_capability_device_type: 9
- msix_support: true
- msix_pba_bar: 4
- msix_table_bar: 2
- msix_table_size: 2048
- power_mgmt_capability: true
- pasid_supported: false
- ext_fmt_field_supported: true
- extended_tag_supported: true
- link_port_number: 0
- pri_supported: true
- rber_supported: true
- slot_and_root_status_msi_idx: 0
- tag_10bit_completer_supported: true
- tag_10bit_requester_supported: true
- aspm_optionality_compliant: true



* indicates parameter values that are fixed and cannot be overridden.

Example

```
"smmu3testengine/smmute": {
  "device": 0,
  "function": 0,
}
```

11.6 Root port parameters

The following root port parameters are available.

Table 11-3: Root port parameters

Parameter	Description	Type	Range	Default value
device_number	Device number on this bus.	int	0-31	0
device_id	PCI device ID.	int	0-0xFFFFE	0xdef
vendor_id	PCI vendor ID.	int	0-0xFFFFE	0x13B5
acs_supported	Access control services supported.	bool	-	false
aspm_optionality_compliant	Enable ASPM optionality compliance.	bool	-	true
ext_fmt_field_supported	Enable extended format field support.	bool	-	true
extended_tag_supported	Extended tag field support.	bool	-	true
link_port_number	Port number for PCIe link.	int	0-255	0
pcie_version	PCIe version, bits[3:0] in capabilities register. 1 is PCIe 3.0. 2 is PCIe 4.0.	int	1-15	2
rber_supported	Enable role-based error reporting.	bool	-	true
slot_and_root_status_msi_idx	MSI index for reporting slot and root status changes.	int	0-2047	0
tag_10bit_completer_supported	Enable 10-bit tag completer support.	bool	-	true
tag_10bit_requester_supported	Enable 10-bit tag requester support.	bool	-	true

Example

```
"rootport/rootport0": {
  "device_number": 5,
  "downstream": {
    "ahci/ahci1": {
      "device": 0,
      "function": 0,
      "image_path": "/path/to/image"
    }
  }
}
```

}

11.7 Switch parameters

This topic lists the switch parameters with their default values and gives an example.

Table 11-4: Switch parameters

Parameter	Description	Type	Range	Default value
device_number	Device number on this bus.	int	0-31	0
acs_supported	Access control services supported.	bool	-	false

Example

```
"switch/switch0": {
    "device_number": 0,
    "—_downstream_10": {
        "ahci/ahci4": {...}
    },
    "—_downstream_20": {
        "ahci/ahci2": {...}
    }
}
```

11.8 Root bridge parameters

The term *root bridge* is taken from the PCI Firmware Specification Revision 3.x. It refers to an abstraction for the platform's PCI config and memory I/O access mechanism. A root bridge can be used to create multiple ECAM and MMIO regions within the system's ECAM and MMIO limits.

Table 11-5: Root bridge parameters

Parameter	Description	Type	Range
ecam_start	ECAM base address.	int	Min: <System ECAM base> Max:<System ECAM end - 1MB>
ecam_end_incl	ECAM end address.	int	Min:<System ECAM base + 1MB> Max:<System ECAM end>
mem32_start	32-bit memory window base address.	int	Min:<System MEM32 base> Max:<System MEM32 end>
mem32_end_incl	32-bit memory window end address.	int	Min:<System MEM32 base> Max:<System MEM32 end>
mem64_start	64-bit memory window base address.	int	Min:<System MEM64 base> Max:<System MEM64 end>

Parameter	Description	Type	Range
mem64_end_incl	64-bit memory window base address.	int	Min:<System MEM64 base> Max:<System MEM64 end>
ecam_start_bus_number	ECAM start bus number.	int	Min:<System start bus> Max:<System end bus> Note: Each bus consumes 1MB of memory, so each ECAM should have enough space (ecam_start - ecam_end_incl) to accommodate the total number of buses (start_bus - end_bus) allocated to it.

Example

```

"rootbridge/rb0": {
    "ecam_start": 0x60000000,
    "ecam_end_incl": 0x67ffffff,
    "mem32_start": 0x70000000,
    "mem32_end_incl": 0x73ffffff,
    "mem64_start": 0x500000000,
    "mem64_end_incl": 0x67ffffff,
    "ecam_start_bus_number": 0x0,

    "_downstream_": {
        "ahci/ahci0": {
            "device": 30,
            "function": 0,
            "image_path": "",
            "express_capability_device_type": 9
        },
        "ahci/ahci1": {
            "device": 31,
            "function": 0,
            "express_capability_device_type": 9,
            "image_path": ""
        }
    }
}

```

11.9 Exerciser

Exerciser is a PCIe device used to generate various stimuli that can help test PCIe/SMMU integration in the system.

Table 11-6: Default values for endpoint parameters

Endpoint parameter	Default value
device	0
function	0
vendor_id	0x13B5
device_id	0xED01
base_class	0xED

Endpoint parameter	Default value
sub_class	0
bar0_64bit	false
bar1_64bit	false
bar2_64bit	false
bar3_64bit	false
bar4_64bit	false
bar0_log2_size	12
bar1_log2_size	14
bar2_log2_size	15
bar3_log2_size	0
bar4_log2_size	0
bar5_log2_size	12
uses_interrupt	true
interrupt_pin_index	1
multi_function	false
pcie_version	2
prog_iface	0
rev_id	0
subsys_id	0
subsys_vendor_id	0x13B5
express_capability_device_type	0
msix_support	true
msix_pba_bar	4
msix_table_bar	2
msix_table_size	2048
power_mgmt_capability	true
pasid_supported	false
ext_fmt_field_supported	true
extended_tag_supported	true
link_port_number	0
pri_supported	true
rber_supported	true
slot_and_root_status_msi_idx	0
tag_10bit_completer_supported	true
tag_10bit_requester_supported	true
aspm_optionality_compliant	true
ats_supported	true
error_injection_supported	false
aer_supported	false

Exerciser parameter and default value

memory_bar

BAR index to be used for exerciser memory access. Type: int, min: 0, max: 5, default: 1.

Example

```
"exerciser/exerciser0": {
  "device": 1,
  "function": 0,
  "interrupt_pin_index": 1,
  "memory_bar": 3,
  "express_capability_device_type": 0
}
```

11.10 Example hierarchy for a single ECAM configuration

The absence of a root bridge before the beginning of the device hierarchy implies a single ECAM configuration.

```
{
  "ahci/ahci0": {
    "device": 30,
    "function": 0,
    "image_path": "",
    "express_capability_device_type": 9
  },
  "smmuv3testengine/smmuv3testengine3": {
    "device": 15,
    "function": 0,
    "express_capability_device_type": 9
  },
  "rootport/rootport0": {
    "device_number": 2,
    "__downstream__": {
      "ahci/ahci1": {
        "device": 0,
        "function": 0,
        "image_path": "",
        "multi_function": true,
      },
      "ahci/ahci2": {
        "device": 0,
        "function": 1,
        "image_path": ""
      }
    }
  },
  "rootport/rootport1": {
    "device_number": 5,
    "__downstream__": {
      "ahci/ahci3": {
        "device": 0,
        "function": 0,
        "image_path": ""
      }
    }
  },
  "rootport/rootport2": {
    "device_number": 6,
    "__downstream__": {
      "switch/switch0": {
```

```

        "device_number": 0,
        "___downstream_10": {
            "ahci/ahci4": {}
        },
        "___downstream_20": {
            "ahci/ahci5": {}
        }
    }
}
}
}
}
}

```

11.11 Example hierarchy for two ECAM configuration

This example shows the JSON file for two ECAMs.

```

{
  "rootbridge/rb0": {
    "ecam_start": 0x60000000,
    "ecam_end_incl": 0x67ffffff,
    "mem32_start": 0x70000000,
    "mem32_end_incl": 0x73ffffff,
    "mem64_start": 0x500000000,
    "mem64_end_incl": 0x67ffffff,
    "ecam_start_bus_number": 0x0,

    "___downstream___": {
      "ahci/ahci0": {
        "device": 30,
        "function": 0,
        "image_path": "",
        "express_capability_device_type": 9
      },
      "ahci/ahci1": {
        "device": 31,
        "function": 0,
        "express_capability_device_type": 9,
        "image_path": ""
      }
    }
  },
  "rootbridge/rb1": {
    "ecam_start": 0x68000000,
    "ecam_end_incl": 0x6ffffff,
    "mem32_start": 0x74000000,
    "mem32_end_incl": 0x77ffffff,
    "mem64_start": 0x680000000,
    "mem64_end_incl": 0x7ffffff,
    "ecam_start_bus_number": 0x1,

    "___downstream___": {
      "ahci/ahci0": {
        "device": 2,
        "function": 0,
        "image_path": "/path/to/image",
        "express_capability_device_type": 9
      }
    }
  }
}
}

```

12. Base Platform FVPs

This chapter describes the Base Platform FVPs contained in the FVP Standard Library and the Arm Architecture Envelope Models (AEM) FVPs.

For the Base Platform memory map, see [Base Platform memory map](#) in the *Fast Models Reference Guide*.

12.1 FVP_Base_AEMv8R

List of instances in FVP_Base_AEMv8R.

FVP_Base_AEMv8R instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBUSSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave
Type: [PVBusSlave](#).

bp.dram_limiter
Type: [PVBusMapper](#).

bp.dummy_local_dap_rom
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave
Type: [PVBusSlave](#).

bp.dummy_ram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_ram.bus_slave
Type: [PVBusSlave](#).

bp.dummy_usb
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.ns_dram.bus_slave`

Type: `PVBusSlave`.

`bp.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`bp.pl011_uart0.busslave`

Type: `PVBusSlave`.

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARMAEMv8-R Cluster CT model.

Type: [cluster_ARMAEMv8-R_MP](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.MMAPType: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARMAEMv8-R MP CT model.

Type: [ARMAEMv8-R_MP](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu1**

ARMAEMv8-R MP CT model.

Type: [ARMAEMv8-R_MP](#).**cluster0.cpu1.UTLB**

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu1.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu1.l1dcache**

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARMAEMv8-R MP CT model.

Type: [ARMAEMv8-R_MP](#).

cluster0.cpu2.UTLB

Type: [TLB](#).

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3

ARMAEMv8-R MP CT model.

Type: [ARMAEMv8-R_MP](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

`cluster0.cpu3.l1dcache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.cpu3.l1icache`
PV Cache.
Type: `PVCache`.

`cluster0.cpu3.l1icache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.ext_bus`
Type: `PVBusLogger`.

`cluster0.ext_bus.mapper`
Type: `PVBusMapper`.

`cluster0.gic_cpuif_decoder_cluster`
Type: `GICv3CPUInterfaceDecoder`.

`cluster0.l2_cache`
PV Cache.
Type: `PVCache`.

`cluster0.l2_cache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[10]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[11]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[12]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[13]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[14]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[15]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[16]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[1]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[2]`
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_3

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0
Type: [PVBusMaster](#).

pctl.utility_bus1
Type: [PVBusMaster](#).

pctl.utility_bus2
Type: [PVBusMaster](#).

pctl.utility_bus3
Type: [PVBusMaster](#).

12.2 FVP_Base_AEMvA-AEMvA

List of instances in FVP_Base_AEMvA-AEMvA.

FVP_Base_AEMvA-AEMvA instances

address_map_terminator
Type: [PVBusMapper](#).

bp
Peripherals and address map for the Base Platform.
Type: [BasePlatformPeripherals](#).

bp.Timer_0_1
ARM Dual-Timer Module(SP804).
Type: [SP804_Timer](#).

bp.Timer_0_1.busslave
Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.counter0
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBUSSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM AEMvA Cluster CT model.

Type: [Cluster_ARM_AEM-A_MP](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu1**

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).**cluster0.cpu1.UTLB**Type: [TLB](#).

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2**

ARM AEM-A MP CT model.

Type: `ARM_AEM-A_MP`.**cluster0.cpu2.UTLB**Type: `TLB`.**cluster0.cpu2.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu2.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu2.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM AEM-A MP CT model.

Type: `ARM_AEM-A_MP`.

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu3.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.l1dcache

PV Cache.

Type: PVPcache.

cluster0.cpu3.l1dcache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu3.l1icache

PV Cache.

Type: PVPcache.

cluster0.cpu3.l1icache.upstream[0]

Type: PVPBusSlave.

cluster0.ext_bus

Type: PVPBusLogger.

cluster0.ext_bus.mapper

Type: PVPBusMapper.

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: PVPcache.

cluster0.l2_cache.upstream[0]

Type: PVPBusSlave.

cluster0.l2_cache.upstream[10]

Type: PVPBusSlave.

cluster0.l2_cache.upstream[11]

Type: PVPBusSlave.

cluster0.l2_cache.upstream[12]

Type: PVPBusSlave.

cluster0.l2_cache.upstream[13]

Type: PVPBusSlave.

cluster0.l2_cache.upstream[14]

Type: PVPBusSlave.

cluster0.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

cluster1

ARM AEMvA Cluster CT model.

Type: [Cluster_ARM_AEM-A_MP](#).

cluster1.AMU

Type: [PVBusLogger](#).

cluster1.AMU.mapper

Type: [PVBusMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: dsu.

cluster1.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.acp_mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu1

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster1.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster1.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster1.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster1.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster1.cpu2

ARM AEM-A MP CT model.

Type: `ARM_AEM-A_MP`.

cluster1.cpu2.UTLB

Type: `TLB`.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster1.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster1.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster1.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster1.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster1.cpu3

ARM AEM-A MP CT model.

Type: `ARM_AEM-A_MP`.

cluster1.cpu3.UTLB

Type: `TLB`.

cluster1.cpu3.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster1.cpu3.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster1.cpu3.l1dcache
PV Cache.
Type: pVCache.

cluster1.cpu3.l1dcache.upstream[0]
Type: pVBusSlave.

cluster1.cpu3.l1icache
PV Cache.
Type: pVCache.

cluster1.cpu3.l1icache.upstream[0]
Type: pVBusSlave.

cluster1.ext_bus
Type: pVBusLogger.

cluster1.ext_bus.mapper
Type: pVBusMapper.

cluster1.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster1.l2_cache
PV Cache.
Type: pVCache.

cluster1.l2_cache.upstream[0]
Type: pVBusSlave.

cluster1.l2_cache.upstream[10]
Type: pVBusSlave.

cluster1.l2_cache.upstream[11]
Type: pVBusSlave.

cluster1.l2_cache.upstream[12]
Type: pVBusSlave.

cluster1.l2_cache.upstream[13]
Type: pVBusSlave.

cluster1.l2_cache.upstream[14]
Type: pVBusSlave.

cluster1.l2_cache.upstream[15]
Type: pVBusSlave.

cluster1.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster1.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster1_labeller
Type: [Labeller](#).

cluster1_labeller.pvbusmodifier
Type: [PVBUSMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBUSLogger](#).

dapmemlogger.mapper
Type: [PVBUSMapper](#).

elfloader
ELF loader component.
Type: [ElfLoader](#).

elfloader.pvbus_busmaster
Type: [PVBUSMaster](#).

gic_distributor
GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_1

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_2

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_3

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1_1

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1_2

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1_3

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.3 FVP_Base_Cortex-A32x1

List of instances in FVP_Base_Cortex-A32x1.

FVP_Base_Cortex-A32x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A32 Cluster CT model.

Type: `Cluster_ARM_Cortex-A32`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.acp_mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A32 CT model.

Type: `ARM_Cortex-A32`.**cluster0.cpu0.S1TLB**Type: `TLB`.**cluster0.cpu0.S2TLB**Type: `TLB`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.4 FVP_Base_Cortex-A32x2

List of instances in FVP_Base_Cortex-A32x2.

FVP_Base_Cortex-A32x2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.psram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.psram.bus_slave

Type: `PVBusSlave`.

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

bp.refcounter.pvbus_control_s[0]

Type: `PVBusSlave`.

bp.refcounter.pvbus_read_s[0]

Type: `PVBusSlave`.

bp.reset_or

Or Gate.

Type: `OrGate`.

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rl_dram.bus_slave

Type: `PVBusSlave`.

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rt_dram.bus_slave

Type: `PVBusSlave`.

bp.s_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.s_dram.bus_slave

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtioblockdevice.dma_master

Type: `PVBusMaster`.

bp.virtioblockdevice_labeller

Type: `Labeller`.

bp.virtioblockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtiop9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtiop9device.mmio_slave

Type: `PVBusSlave`.

bp.virtiop9device.virtio_master

Type: `PVBusMaster`.

bp.virtiop9device_labeller

Type: `Labeller`.

bp.virtiop9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A32 Cluster CT model.

Type: [Cluster_ARM_Cortex-A32](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A32 CT model.

Type: [ARM_Cortex-A32](#).

cluster0.cpu0.S1TLB

Type: TLB.

cluster0.cpu0.S2TLB

Type: TLB.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A32 CT model.

Type: [ARM_Cortex-A32](#).

cluster0.cpu1.S1TLB

Type: TLB.

cluster0.cpu1.S2TLB

Type: TLB.

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu1.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu1.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0
Type: [PVBusMaster](#).

pctl.utility_bus1
Type: [PVBusMaster](#).

pctl.utility_bus2
Type: [PVBusMaster](#).

pctl.utility_bus3
Type: [PVBusMaster](#).

12.5 FVP_Base_Cortex-A32x4

List of instances in FVP_Base_Cortex-A32x4.

FVP_Base_Cortex-A32x4 instances

address_map_terminator
Type: [PVBusMapper](#).

bp
Peripherals and address map for the Base Platform.
Type: [BasePlatformPeripherals](#).

bp.Timer_0_1
ARM Dual-Timer Module(SP804).
Type: [SP804_Timer](#).

bp.Timer_0_1.busslave
Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.counter0
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A32 Cluster CT model.

Type: [Cluster_ARM_Cortex-A32](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A32 CT model.

Type: [ARM_Cortex-A32](#).

cluster0.cpu0.S1TLB

Type: TLB.

cluster0.cpu0.S2TLB

Type: TLB.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A32 CT model.

Type: [ARM_Cortex-A32](#).

cluster0.cpu1.S1TLB

Type: TLB.

cluster0.cpu1.S2TLB

Type: TLB.

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu1.l1dcache.upstream[0]

Type: PVBusslave.

cluster0.cpu1.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu1.l1icache.upstream[0]

Type: PVBusslave.

cluster0.cpu2

ARM Cortex-A32 CT model.

Type: ARM_Cortex-A32.

cluster0.cpu2.S1TLB

Type: TLB.

cluster0.cpu2.S2TLB

Type: TLB.

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1dcache.upstream[0]

Type: PVBusslave.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-A32 CT model.

Type: `ARM_Cortex-A32`.**cluster0.cpu3.S1TLB**

Type: TLB.

cluster0.cpu3.S2TLB

Type: TLB.

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu3.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.ext_bus**Type: `PVBusLogger`.**cluster0.ext_bus.mapper**Type: `PVBusMapper`.**cluster0.gic_cpuif_decoder_cluster**Type: `GICv3CPUInterfaceDecoder`.**cluster0.l2_cache**

PV Cache.

Type: `PVCache`.

cluster0.12_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.12_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifierType: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_tl**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#)

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.6 FVP_Base_Cortex-A34x1

List of instances in `FVP_Base_Cortex-A34x1`.

FVP_Base_Cortex-A34x1 instances

address_map_terminator

Type: `PVBusMapper`.

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A34 Cluster CT model.

Type: [cluster_ARM_Cortex-A34](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A34 CT model.

Type: [ARM_Cortex-A34](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

`cluster0.cpu0.l1dcache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.cpu0.l1icache`
PV Cache.
Type: `PVCache`.

`cluster0.cpu0.l1icache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.ext_bus`
Type: `PVBusLogger`.

`cluster0.ext_bus.mapper`
Type: `PVBusMapper`.

`cluster0.gic_cpuif_decoder_cluster`
Type: `GICv3CPUInterfaceDecoder`.

`cluster0.l2_cache`
PV Cache.
Type: `PVCache`.

`cluster0.l2_cache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[10]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[11]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[12]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[13]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[14]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[15]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[16]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[1]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[2]`
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[3]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBusLogger](#).

dapmemlogger.mapper
Type: [PVBusMapper](#).

elfloader
ELF loader component.
Type: [ElfLoader](#).

elfloader.pvbus_busmaster
Type: [PVBusMaster](#).

gic_distributor
GIC Interrupt Redistribution Infrastructure component.
Type: [GIC_IRI](#).

gic_distributor.rd_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).**pctl.utility_bus2**Type: [PVBusMaster](#).

pctl.utility_bus3Type: [PVBusMaster](#).

12.7 FVP_Base_Cortex-A34x2

List of instances in FVP_Base_Cortex-A34x2.

FVP_Base_Cortex-A34x2 instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_2_3.busslave**Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A34 Cluster CT model.

Type: [Cluster_ARM_Cortex-A34](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A34 CT model.

Type: [ARM_Cortex-A34](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A34 CT model.

Type: [ARM_Cortex-A34](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.ext_bus

Type: `PVBusLogger`.

cluster0.ext_bus.mapper

Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster

Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache

PV Cache.

Type: `PVCache`.

cluster0.l2_cache.upstream[0]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[13]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[14]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[15]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[16]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[1]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[2]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[3]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[4]

Type: `PVBusSlave`.

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_1

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.8 FVP_Base_Cortex-A34x4

List of instances in FVP_Base_Cortex-A34x4.

FVP_Base_Cortex-A34x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A34 Cluster CT model.

Type: [Cluster_ARM_Cortex-A34](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A34 CT model.

Type: [ARM_Cortex-A34](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A34 CT model.

Type: [ARM_Cortex-A34](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2**

ARM Cortex-A34 CT model.

Type: `ARM_Cortex-A34`.**cluster0.cpu2.UTLB**Type: `TLB`.**cluster0.cpu2.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu2.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu2.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-A34 CT model.

Type: `ARM_Cortex-A34`.**cluster0.cpu3.UTLB**Type: `TLB`.**cluster0.cpu3.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu3.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_3

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).

pctl.utility_bus1
Type: [PVBusMaster](#).

pctl.utility_bus2
Type: [PVBusMaster](#).

pctl.utility_bus3
Type: [PVBusMaster](#).

12.9 FVP_Base_Cortex-A35x1

List of instances in FVP_Base_Cortex-A35x1.

FVP_Base_Cortex-A35x1 instances

address_map_terminator
Type: [PVBusMapper](#).

bp
Peripherals and address map for the Base Platform.
Type: [BasePlatformPeripherals](#).

bp.Timer_0_1
ARM Dual-Timer Module(SP804).
Type: [SP804_Timer](#).

bp.Timer_0_1.busslave
Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.counter0
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_0_1.counter1
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.ns_dram.bus_slave`

Type: `PVBusSlave`.

`bp.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`bp.pl011_uart0.busslave`

Type: `PVBusSlave`.

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A35 Cluster CT model.

Type: [cluster_ARM_Cortex-A35](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A35 CT model.

Type: [ARM_Cortex-A35](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.12_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.12_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifierType: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.10 FVP_Base_Cortex-A35x2

List of instances in FVP_Base_Cortex-A35x2.

FVP_Base_Cortex-A35x2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A35 Cluster CT model.

Type: `Cluster_ARM_Cortex-A35`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.acp_mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A35 CT model.

Type: `ARM_Cortex-A35`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A35 CT model.

Type: [ARM_Cortex-A35](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuiif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.12_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_1

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.11 FVP_Base_Cortex-A35x4

List of instances in FVP_Base_Cortex-A35x4.

FVP_Base_Cortex-A35x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10Type: [PVBusSlave](#).**bp.ap_refclk.busbase11**Type: [PVBusSlave](#).**bp.ap_refclk.busbase12**Type: [PVBusSlave](#).**bp.ap_refclk.busbase13**Type: [PVBusSlave](#).**bp.ap_refclk.busbase14**Type: [PVBusSlave](#).**bp.ap_refclk.busbase15**Type: [PVBusSlave](#).**bp.ap_refclk.busbase2**Type: [PVBusSlave](#).**bp.ap_refclk.busbase3**Type: [PVBusSlave](#).**bp.ap_refclk.busbase4**Type: [PVBusSlave](#).**bp.ap_refclk.busbase5**Type: [PVBusSlave](#).**bp.ap_refclk.busbase6**Type: [PVBusSlave](#).**bp.ap_refclk.busbase7**Type: [PVBusSlave](#).**bp.ap_refclk.busbase8**Type: [PVBusSlave](#).**bp.ap_refclk.busbase9**Type: [PVBusSlave](#).**bp.ap_refclk.busctlbase**Type: [PVBusSlave](#).**bp.audioout**

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).**bp.clock100Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.pl180_mci**

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).**bp.pl180_mci.busslave**Type: [PVBusSlave](#).**bp.ps2keyboard**

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).**bp.ps2keyboard.ps2_clocktimer**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**bp.ps2keyboard.ps2_clocktimer.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**bp.ps2keyboard.ps2_clocktimer.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**bp.ps2keyboard.ps2_clocktimer.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**bp.ps2mouse**

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A35 Cluster CT model.

Type: [cluster_ARM_Cortex-A35](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.acp_mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu1

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu1.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu2

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu2.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu3

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuiif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_3

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.12 FVP_Base_Cortex-A510

List of instances in FVP_Base_Cortex-A510.

FVP_Base_Cortex-A510 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave
Type: [PVBusSlave](#).

bp.dmc_phy
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dmc_phy.bus_slave
Type: [PVBusSlave](#).

bp.dram_limiter
Type: [PVBusMapper](#).

bp.dummy_local_dap_rom
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave
Type: [PVBusSlave](#).

bp.dummy_ram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_ram.bus_slave
Type: [PVBusSlave](#).

bp.dummy_usb
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbType: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmb**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLC](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A510Cluster CT model.

Type: [Cluster_ARM_Cortex-A510](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core10

Type: PPUv1.

cluster0.DSU.PPU_core10.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core11

Type: PPUv1.

cluster0.DSU.PPU_core11.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core6

Type: PPUv1.

cluster0.DSU.PPU_core6.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core7

Type: PPUv1.

cluster0.DSU.PPU_core7.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core8

Type: PPUv1.

cluster0.DSU.PPU_core8.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core9

Type: PPUv1.

cluster0.DSU.PPU_core9.busslave

Type: PVBusSlave.

cluster0.DSU.l3_flusher

Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVCache.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[10]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[11]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[12]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[13]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[14]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[15]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1dcache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu1.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1icache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu1.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l2cache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu1.l2cache.upstream[1]
Type: [PVBusSlave](#).

cluster0.cpu10
ARM Cortex-A510 CT model.
Type: [ARM_Cortex-A510](#).

cluster0.cpu10.UTLB
Type: [TLB](#).

cluster0.cpu10.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu10.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu10.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu10.l1dcache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu10.l1icache
PV Cache.

Type: `PVCache`.

cluster0.cpu10.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu10.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu10.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu10.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu11
ARM Cortex-A510 CT model.
Type: `ARM_Cortex-A510`.

cluster0.cpu11.UTLB
Type: `TLB`.

cluster0.cpu11.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu11.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu11.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu2
ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu3.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu4

ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu4.UTLB

Type: [TLB](#).

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu4.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu6

ARM Cortex-A510 CT model.

Type: [ARM_Cortex-A510](#).

cluster0.cpu6.UTLB

Type: TLB.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu6.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu7**

ARM Cortex-A510 CT model.

Type: `ARM_Cortex-A510`.**cluster0.cpu7.UTLB**Type: `TLB`.**cluster0.cpu7.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu7.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu7.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu8

ARM Cortex-A510 CT model.

Type: `ARM_Cortex-A510`.

cluster0.cpu8.UTLB

Type: `TLB`.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu8.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu8.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu8.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu8.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu8.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu9

ARM Cortex-A510 CT model.

Type: `ARM_Cortex-A510`.

cluster0.cpu9.UTLB

Type: `TLB`.

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu9.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu9.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_11

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_11_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_2_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_3_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_4

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_4_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_5

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.13 FVP_Base_Cortex-A510+Cortex-A710

List of instances in FVP_Base_Cortex-A510+Cortex-A710.

FVP_Base_Cortex-A510+Cortex-A710 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.coresight_modifier

Type: `PVBusMapper`.

bp.dmc

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc.bus_slave

Type: `PVBusSlave`.

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc_phy.bus_slave

Type: `PVBusSlave`.

bp.dram_limiter

Type: `PVBusMapper`.

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_local_dap_rom.bus_slave

Type: `PVBusSlave`.

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_ram.bus_slave

Type: `PVBusSlave`.

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_usb.bus_slave

Type: `PVBusSlave`.

bp.exclusive_monitor

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

bp.exclusive_monitor.bus_mapper

Type: `PVBusMapper`.

bp.fixed_security_map

Type: `PVBusMapper`.

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash0.map

Type: `PVBusMapper`.

bp.flash0.mbs

Type: `PVBusSlave`.

bp.flash0.rmbs

Type: `PVBusSlave`.

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash1.map

Type: `PVBusMapper`.

bp.flash1.mbs

Type: `PVBusSlave`.

bp.flash1.rmbs

Type: `PVBusSlave`.

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.generic_watchdog

ARM Generic Watchdog.

Type: `MemoryMappedGenericWatchdog`.

bp.generic_watchdog.busctlbase

Type: `PVBusSlave`.

bp.generic_watchdog.busrefbase

Type: `PVBusSlave`.

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCD`.

bp.hdlcd0.busmaster

Type: `PVBusMaster`.

bp.hdlcd0.busslave

Type: `PVBusSlave`.

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slaveType: [PVBusSlave](#).**bp.pl011_uart0**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart0.busslave**Type: [PVBusSlave](#).**bp.pl011_uart0.clk_divider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.pl011_uart0.timer**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**bp.pl011_uart0.timer.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**bp.pl011_uart0.timer.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**bp.pl011_uart0.timer.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**bp.pl011_uart1**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart1.busslave**Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl1111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl1111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).**bp.rl_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.rl_dram.bus_slave**Type: [PVBusSlave](#).**bp.rt_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.rt_dram.bus_slave**Type: [PVBusSlave](#).**bp.s_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.s_dram.bus_slave**Type: [PVBusSlave](#).**bp.secureDRAM**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.secureDRAM.bus_slave**Type: [PVBusSlave](#).**bp.secureSRAM**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.secureSRAM.bus_slave**Type: [PVBusSlave](#).**bp.secureSRAM_limiter**Type: [PVBusMapper](#).**bp.secureflash**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.secureflash.map**Type: [PVBusMapper](#).**bp.secureflash.mbs**Type: [PVBusSlave](#).**bp.secureflash.rmbs**Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBusSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: filter0.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: filter1.

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: filter2.

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: filter3.

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [VE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A510, Cortex-A710 CT heterogeneous cluster model.

Type: `cluster_ARM_Cortex510_CortexA710_Heterogeneous`.

cluster0.AMU

Type: `PVBusLogger`.

cluster0.AMU.mapper

Type: `PVBusMapper`.

cluster0.DAP

Type: `PVBusLogger`.

cluster0.DAP.mapper

Type: `PVBusMapper`.

cluster0.DSU

Type: `DSU`.

cluster0.DSU.PPU_cluster

Type: `PPUv1`.

cluster0.DSU.PPU_cluster.busslave

Type: `PVBusSlave`.

cluster0.DSU.PPU_core0

Type: `PPUv1`.

cluster0.DSU.PPU_core0.busslave

Type: `PVBusSlave`.

cluster0.DSU.PPU_core1

Type: `PPUv1`.

cluster0.DSU.PPU_core1.busslave

Type: `PVBusSlave`.

cluster0.DSU.mpam_busslave

Type: `PVBusSlave`.

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.

cluster0.DSU.shared_cache.upstream[0]

Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[1]

Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[2]

Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[3]

Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[4]
Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[5]
Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]
Type: [PVBusSlave](#).

cluster0.MMAP
Type: [PVBusLogger](#).

cluster0.MMAP.mapper
Type: [PVBusMapper](#).

cluster0.ext_bus
Type: [PVBusLogger](#).

cluster0.ext_bus.mapper
Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.subcluster0
ARM Cortex-A510 CT Model in Heterogeneous Cluster.
Type: [Subcluster_ARM_Cortex-A510](#).

cluster0.subcluster0.cpu0
ARM Cortex-A510 CT model.
Type: [ARM_Cortex-A510](#).

cluster0.subcluster0.cpu0.UTLB
Type: TLB.

cluster0.subcluster0.cpu0.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.subcluster0.cpu0.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.subcluster0.cpu0.l1dcache
PV Cache.
Type: [PVCache](#).

cluster0.subcluster0.cpu0.l1dcache.upstream[0]
Type: [PVBusSlave](#).

cluster0.subcluster0.cpu0.l1icache
PV Cache.
Type: [PVCache](#).

cluster0.subcluster0.cpu0.l1icache.upstream[0]
Type: [PVBusSlave](#).

cluster0.subcluster0.cpu0.l2cache

PV Cache.

Type: `PVCache`.**cluster0.subcluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.subcluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.subcluster1**

ARM Cortex-A710 CT Model in Heterogeneous Cluster.

Type: `subcluster_ARM_Cortex-A710`.**cluster0.subcluster1.cpu0**

ARM Cortex-A710 CT model.

Type: `ARM_Cortex-A710`.**cluster0.subcluster1.cpu0.UTLB**Type: `TLB`.**cluster0.subcluster1.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.subcluster1.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.subcluster1.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.subcluster1.cpu0.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.subcluster1.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.subcluster1.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.subcluster1.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.subcluster1.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.subcluster1.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0_labeller**Type: `Labeller`.

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_11

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_11_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_12

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_12_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_13
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_13_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_14
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_14_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_15
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_15_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_16
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_16_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_17
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_17_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_18
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_18_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_19
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_19_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_20
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_20_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_21
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_21_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_8
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_9
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0
Type: GICv3Redistributor.

gic_distributor.rd_tl
Type: GICv3Distributor.

pctl
Base Platforms Power Controller.
Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.14 FVP_Base_Cortex-A520

List of instances in FVP_Base_Cortex-A520.

FVP_Base_Cortex-A520 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A520 Cluster CT model.

Type: [Cluster_ARM_Cortex-A520](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUv1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBUSSlave](#).

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core10

Type: PPUv1.

cluster0.DSU.PPU_core10.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core11

Type: PPUv1.

cluster0.DSU.PPU_core11.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core12

Type: PPUv1.

cluster0.DSU.PPU_core12.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core13

Type: PPUv1.

cluster0.DSU.PPU_core13.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core6

Type: [PPUv1](#).

cluster0.DSU.PPU_core6.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core7

Type: [PPUv1](#).

cluster0.DSU.PPU_core7.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core8

Type: [PPUv1](#).

cluster0.DSU.PPU_core8.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core9

Type: [PPUv1](#).

cluster0.DSU.PPU_core9.busslave

Type: [PVBusSlave](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[26]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[27]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[28]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[29]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[30]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[5]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[6]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[7]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[8]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[9]**Type: [PVBUSSlave](#).**cluster0.DSU.utility_slave[0]**Type: [PVBUSSlave](#).**cluster0.MMAP**Type: [PVBUSLogger](#).**cluster0.MMAP.mapper**Type: [PVBUSMapper](#).**cluster0.cpu0**

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-A520 CT model.

Type: `ARM_Cortex-A520`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu10**

ARM Cortex-A520 CT model.

Type: `ARM_Cortex-A520`.**cluster0.cpu10.UTLB**Type: `TLB`.

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu10.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu10.l1dcache

PV Cache.

Type: pVCache.

cluster0.cpu10.l1dcache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu10.l1icache**

PV Cache.

Type: pVCache.

cluster0.cpu10.l1icache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu10.l2cache**

PV Cache.

Type: pVCache.

cluster0.cpu10.l2cache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu10.l2cache.upstream[1]**Type: [PVBUSSlave](#).**cluster0.cpu11**

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).**cluster0.cpu11.UTLB**

Type: TLB.

cluster0.cpu11.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu11.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu11.l1dcache

PV Cache.

Type: pVCache.

cluster0.cpu11.l1dcache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu11.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu11.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu12
ARM Cortex-A520 CT model.
Type: `ARM_Cortex-A520`.

cluster0.cpu12.UTLB
Type: `TLB`.

cluster0.cpu12.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu12.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu12.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu12.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu12.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu12.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu12.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu12.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu12.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu13
ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).

cluster0.cpu13.UTLB

Type: TLB.

cluster0.cpu13.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu13.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu13.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l1dcache.upstream[0]

Type: [PVBusslave](#).

cluster0.cpu13.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l1icache.upstream[0]

Type: [PVBusslave](#).

cluster0.cpu13.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l2cache.upstream[0]

Type: [PVBusslave](#).

cluster0.cpu13.l2cache.upstream[1]

Type: [PVBusslave](#).

cluster0.cpu2

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu3

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).

cluster0.cpu4.UTLB

Type: TLB.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu4.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu6**

ARM Cortex-A520 CT model.

Type: `ARM_Cortex-A520`.**cluster0.cpu6.UTLB**Type: `TLB`.**cluster0.cpu6.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu6.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu6.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu6.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu6.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu7

ARM Cortex-A520 CT model.

Type: `ARM_Cortex-A520`.

cluster0.cpu7.UTLB

Type: `TLB`.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu7.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu7.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu8

ARM Cortex-A520 CT model.

Type: `ARM_Cortex-A520`.

cluster0.cpu8.UTLB

Type: `TLB`.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu8.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu8.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu9

ARM Cortex-A520 CT model.

Type: [ARM_Cortex-A520](#).

cluster0.cpu9.UTLB

Type: [TLB](#).

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu9.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu9.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu9.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu9.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu9.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu9.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**Type: [GICv3CPUInterfaceDecoder](#).**cluster0_labeller**Type: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_11
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_11_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_12
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_12_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_13
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_13_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.15 FVP_Base_Cortex-A53x1

List of instances in FVP_Base_Cortex-A53x1.

FVP_Base_Cortex-A53x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: filter0.

bp.tzc_400.filter0.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter1
Type: filter1.

bp.tzc_400.filter1.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A53 Cluster CT model.

Type: [Cluster_ARM_Cortex-A53](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.acp_mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.ext_bus

Type: PVBusLogger.

cluster0.ext_bus.mapper

Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: PVPcache.

cluster0.l2_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifierType: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.16 FVP_Base_Cortex-A53x2

List of instances in FVP_Base_Cortex-A53x2.

FVP_Base_Cortex-A53x2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PvBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PvBusMapper](#).

bp.utility_bus_map0

Type: [PvBusMapper](#).

bp.utility_bus_map1

Type: [PvBusMapper](#).

bp.utility_bus_map2

Type: [PvBusMapper](#).

bp.utility_bus_map3

Type: [PvBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PvBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A53 Cluster CT model.

Type: `Cluster_ARM_Cortex-A53`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.acp_mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A53 CT model.

Type: `ARM_Cortex-A53`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuiif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.12_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_1

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.17 FVP_Base_Cortex-A53x4

List of instances in FVP_Base_Cortex-A53x4.

FVP_Base_Cortex-A53x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBusSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A53 Cluster CT model.

Type: [Cluster_ARM_Cortex-A53](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.acp_mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu1

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu1.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu2

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu2.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu3

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuiif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_3

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.18 FVP_Base_Cortex-A55

List of instances in FVP_Base_Cortex-A55.

FVP_Base_Cortex-A55 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbsType: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A55 Cluster CT model.

Type: [cluster_ARM_Cortex-A55](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.13_flusher

Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVPatch.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[10]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[11]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[12]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[13]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[14]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[15]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[16]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[17]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[1]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[2]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[3]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[4]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[5]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A55 CT model.

Type: [ARM_Cortex-A55](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A55 CT model.

Type: [ARM_Cortex-A55](#).**cluster0.cpu1.UTLB**

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: pVCache.

cluster0.cpu1.l1dcache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu1.l1icache**

PV Cache.

Type: pVCache.

cluster0.cpu1.l1icache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu1.l2cache**

PV Cache.

Type: pVCache.

cluster0.cpu1.l2cache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu1.l2cache.upstream[1]**Type: [PVBUSSlave](#).**cluster0.cpu2**

ARM Cortex-A55 CT model.

Type: [ARM_Cortex-A55](#).**cluster0.cpu2.UTLB**

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A55 CT model.

Type: `ARM_Cortex-A55`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu3.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu3.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu4

ARM Cortex-A55 CT model.

Type: [ARM_Cortex-A55](#).

cluster0.cpu4.UTLB

Type: TLB.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu4.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu5

ARM Cortex-A55 CT model.

Type: [ARM_Cortex-A55](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu5.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu5.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu5.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu5.l2cache

PV Cache.

Type: PVCache.

cluster0.cpu5.l2cache.upstream[0]

Type: PVBusSlave.

cluster0.cpu5.l2cache.upstream[1]

Type: PVBusSlave.

cluster0.cpu6

ARM Cortex-A55 CT model.

Type: ARM_Cortex-A55.

cluster0.cpu6.UTLB

Type: TLB.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu6.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu6.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu6.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu6.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu6.l2cache

PV Cache.

Type: [PVCache](#).**cluster0.cpu6.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu6.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu7**

ARM Cortex-A55 CT model.

Type: [ARM_Cortex-A55](#).**cluster0.cpu7.UTLB**Type: [TLB](#).**cluster0.cpu7.dtlb**

TLB - instruction, data or unified.

Type: [Tlbcadi](#).**cluster0.cpu7.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu7.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu7.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu7.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu7.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu7.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu7.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu7.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0_labeller
Type: Labeller.

cluster0_labeller.pvbusmodifier
Type: PVBusMapper.

dapmemlogger
Bus Logger.
Type: PVBusLogger.

dapmemlogger.mapper
Type: PVBusMapper.

elfloader
ELF loader component.
Type: ElfLoader.

elfloader.pvbus_busmaster
Type: PVBusMaster.

gic_distributor
GIC Interrupt Redistribution Infrastructure component.
Type: GIC_IRI.

gic_distributor.rd_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_4

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_5

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBUSSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.19 FVP_Base_Cortex-A55+Cortex-A76

List of instances in FVP_Base_Cortex-A55+Cortex-A76.

FVP_Base_Cortex-A55+Cortex-A76 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBusSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A55_Cortex-A76 Cluster CT model.

Type: [cluster_ARM_Cortex-A55_Cortex-A76](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.13_flusher

Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVPatch.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[1]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[2]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[3]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[4]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[5]

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.ext_bus

Type: PVBusLogger.

cluster0.ext_bus.mapper

Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.subcluster0

ARM Cortex-A55 Cluster CT model.

Type: Subcluster_ARM_Cortex-A55.

cluster0.subcluster0.cpu0

ARM Cortex-A55 CT model.

Type: ARM_Cortex-A55.

cluster0.subcluster0.cpu0.UTLB

Type: TLB.

cluster0.subcluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.subcluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster0.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.subcluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster0.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster0.subcluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster0.cpu0.l2cache

PV Cache.

Type: Pvcache.

cluster0.subcluster0.cpu0.l2cache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster0.cpu0.l2cache.upstream[1]

Type: PVBusSlave.

cluster0.subcluster1

ARM Cortex-A76 Cluster CT model.

Type: Subcluster_ARM_Cortex-A76.

cluster0.subcluster1.cpu0

ARM Cortex-A76 CT model.

Type: ARM_Cortex-A76.

cluster0.subcluster1.cpu0.UTLB

Type: TLB.

cluster0.subcluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster1.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.subcluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster1.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.subcluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.subcluster1.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.subcluster1.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.subcluster1.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.subcluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_8
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_9_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3Type: [PVBusMaster](#).

12.20 FVP_Base_Cortex-A57x1

List of instances in FVP_Base_Cortex-A57x1.

FVP_Base_Cortex-A57x1 instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_2_3.busslave**Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [VEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [virtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [Cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBUSMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_t1

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.21 FVP_Base_Cortex-A57x1-A35x1

List of instances in FVP_Base_Cortex-A57x1-A35x1.

FVP_Base_Cortex-A57x1-A35x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10Type: [PVBUSSlave](#).**bp.ap_refclk.busbase11**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase12**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase13**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase14**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase15**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase2**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase3**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase4**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase5**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase6**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase7**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase8**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase9**Type: [PVBUSSlave](#).**bp.ap_refclk.busctlbase**Type: [PVBUSSlave](#).**bp.audioout**

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).**bp.clock100Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [Cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]
Type: PVBusSlave.

cluster0.l2_cache.upstream[2]
Type: PVBusSlave.

cluster0.l2_cache.upstream[3]
Type: PVBusSlave.

cluster0.l2_cache.upstream[4]
Type: PVBusSlave.

cluster0.l2_cache.upstream[5]
Type: PVBusSlave.

cluster0.l2_cache.upstream[6]
Type: PVBusSlave.

cluster0.l2_cache.upstream[7]
Type: PVBusSlave.

cluster0.l2_cache.upstream[8]
Type: PVBusSlave.

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: AsyncCacheFlushUnit.

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

cluster1

ARM Cortex-A35 Cluster CT model.

Type: Cluster_ARM_Cortex-A35.

cluster1.AMU

Type: [PVBusLogger](#).

cluster1.AMU.mapper

Type: [PVBusMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: dsu.

cluster1.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.acp_mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Cortex-A35 CT model.

Type: [ARM_Cortex-A35](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu0.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache
PV Cache.
Type: pvcache.

cluster1.cpu0.l1dcache.upstream[0]
Type: PVBusSlave.

cluster1.cpu0.l1icache
PV Cache.
Type: pvcache.

cluster1.cpu0.l1icache.upstream[0]
Type: PVBusSlave.

cluster1.ext_bus
Type: PVBusLogger.

cluster1.ext_bus.mapper
Type: PVBusMapper.

cluster1.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster1.l2_cache
PV Cache.
Type: pvcache.

cluster1.l2_cache.upstream[0]
Type: PVBusSlave.

cluster1.l2_cache.upstream[10]
Type: PVBusSlave.

cluster1.l2_cache.upstream[11]
Type: PVBusSlave.

cluster1.l2_cache.upstream[12]
Type: PVBusSlave.

cluster1.l2_cache.upstream[13]
Type: PVBusSlave.

cluster1.l2_cache.upstream[14]
Type: PVBusSlave.

cluster1.l2_cache.upstream[15]
Type: PVBusSlave.

cluster1.l2_cache.upstream[16]
Type: PVBusSlave.

cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.22 FVP_Base_Cortex-A57x1-A53x1

List of instances in FVP_Base_Cortex-A57x1-A53x1.

FVP_Base_Cortex-A57x1-A53x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbsType: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.virtio_rng**

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).**bp.virtio_rng.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice**

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).**bp.virtio_blockdevice.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice_labeller**Type: [Labeller](#).**bp.virtio_blockdevice_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.virtio_p9device**

virtio P9 server.

Type: [VirtioP9Device](#).**bp.virtio_p9device.mmio_slave**Type: [PVBusSlave](#).**bp.virtio_p9device.virtio_master**Type: [PVBusMaster](#).**bp.virtio_p9device_labeller**Type: [Labeller](#).**bp.virtio_p9device_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.vis**

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).**bp.vis.recorder**

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).**bp.vis.recorder.playbackDivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [Cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]
Type: PVBusSlave.

cluster0.l2_cache.upstream[2]
Type: PVBusSlave.

cluster0.l2_cache.upstream[3]
Type: PVBusSlave.

cluster0.l2_cache.upstream[4]
Type: PVBusSlave.

cluster0.l2_cache.upstream[5]
Type: PVBusSlave.

cluster0.l2_cache.upstream[6]
Type: PVBusSlave.

cluster0.l2_cache.upstream[7]
Type: PVBusSlave.

cluster0.l2_cache.upstream[8]
Type: PVBusSlave.

cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

cluster0.l2_flusher
Type: AsyncCacheFlushUnit.

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

cluster1
ARM Cortex-A53 Cluster CT model.
Type: Cluster_ARM_Cortex-A53.

cluster1.AMU
Type: [PVBusLogger](#).

cluster1.AMU.mapper
Type: [PVBusMapper](#).

cluster1.DAP
Type: [PVBusLogger](#).

cluster1.DAP.mapper
Type: [PVBusMapper](#).

cluster1.DSU
Type: dsu.

cluster1.DSU.mpam_busslave
Type: [PVBusSlave](#).

cluster1.MMAP
Type: [PVBusLogger](#).

cluster1.MMAP.mapper
Type: [PVBusMapper](#).

cluster1.acp_mapper
Type: [PVBusMapper](#).

cluster1.cpu0
ARM Cortex-A53 CT model.
Type: [ARM_Cortex-A53](#).

cluster1.cpu0.UTLB
Type: TLB.

cluster1.cpu0.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster1.cpu0.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache
PV Cache.
Type: PVPCache.

cluster1.cpu0.l1dcache.upstream[0]
Type: PVPBusSlave.

cluster1.cpu0.l1icache
PV Cache.
Type: PVPCache.

cluster1.cpu0.l1icache.upstream[0]
Type: PVPBusSlave.

cluster1.ext_bus
Type: PVPBusLogger.

cluster1.ext_bus.mapper
Type: PVPBusMapper.

cluster1.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster1.l2_cache
PV Cache.
Type: PVPCache.

cluster1.l2_cache.upstream[0]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[10]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[11]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[12]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[13]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[14]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[15]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[16]
Type: PVPBusSlave.

cluster1.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.23 FVP_Base_Cortex-A57x2

List of instances in FVP_Base_Cortex-A57x2.

FVP_Base_Cortex-A57x2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbType: [PVBUSSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBUSMapper](#).**bp.flash1.mbs**Type: [PVBUSSlave](#).**bp.flash1.rmb**Type: [PVBUSSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBUSSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBUSSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBUSMaster](#).**bp.hdlcd0.busslave**Type: [PVBUSSlave](#).**bp.hdlcd0.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [Cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.acp_mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-A57 CT model.

Type: ARM_Cortex-A57.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu1

ARM Cortex-A57 CT model.

Type: ARM_Cortex-A57.

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache
PV Cache.
Type: pvcache.

cluster0.cpu1.l1dcache.upstream[0]
Type: PVBusSlave.

cluster0.cpu1.l1icache
PV Cache.
Type: pvcache.

cluster0.cpu1.l1icache.upstream[0]
Type: PVBusSlave.

cluster0.ext_bus
Type: PVBusLogger.

cluster0.ext_bus.mapper
Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: pvcache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0
Type: [PVBusMaster](#).

pctl.utility_bus1
Type: [PVBusMaster](#).

pctl.utility_bus2
Type: [PVBusMaster](#).

pctl.utility_bus3
Type: [PVBusMaster](#).

12.24 FVP_Base_Cortex-A57x2-A35x4

List of instances in FVP_Base_Cortex-A57x2-A35x4.

FVP_Base_Cortex-A57x2-A35x4 instances

address_map_terminator
Type: [PVBusMapper](#).

bp
Peripherals and address map for the Base Platform.
Type: [BasePlatformPeripherals](#).

bp.Timer_0_1
ARM Dual-Timer Module(SP804).
Type: [SP804_Timer](#).

bp.Timer_0_1.busslave
Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.counter0
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.virtio_rng**

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).**bp.virtio_rng.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice**

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).**bp.virtio_blockdevice.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice_labeller**Type: [Labeller](#).**bp.virtio_blockdevice_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.virtio_p9device**

virtio P9 server.

Type: [VirtioP9Device](#).**bp.virtio_p9device.mmio_slave**Type: [PVBusSlave](#).**bp.virtio_p9device.virtio_master**Type: [PVBusMaster](#).**bp.virtio_p9device_labeller**Type: [Labeller](#).**bp.virtio_p9device_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.vis**

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).**bp.vis.recorder**

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).**bp.vis.recorder.playbackDivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [Cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu1**

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).**cluster0.cpu1.UTLB**Type: [TLB](#).

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu1.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.ext_bus
Type: pVBusLogger.

cluster0.ext_bus.mapper
Type: pVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: pVCache.

cluster0.l2_cache.upstream[0]
Type: pVBusSlave.

cluster0.l2_cache.upstream[10]
Type: pVBusSlave.

cluster0.l2_cache.upstream[11]
Type: pVBusSlave.

cluster0.l2_cache.upstream[12]
Type: pVBusSlave.

cluster0.l2_cache.upstream[13]
Type: pVBusSlave.

cluster0.l2_cache.upstream[14]
Type: pVBusSlave.

cluster0.l2_cache.upstream[15]
Type: pVBusSlave.

cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

cluster1

ARM Cortex-A35 Cluster CT model.

Type: [cluster_ARM_Cortex-A35](#).

cluster1.AMU

Type: [PVBusLogger](#).

cluster1.AMU.mapper

Type: [PVBusMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: DSU.

cluster1.DSU.mpam_busslave

Type: PVBusSlave.

cluster1.MMAP

Type: PVBusLogger.

cluster1.MMAP.mapper

Type: PVBusMapper.

cluster1.acp_mapper

Type: PVBusMapper.

cluster1.cpu0

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: PVPcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: PVPcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu2.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu3

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster1.cpu3.UTLB

Type: TLB.

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster1.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.ext_bus
Type: `PVBusLogger`.

cluster1.ext_bus.mapper
Type: `PVBusMapper`.

cluster1.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster1.l2_cache
PV Cache.
Type: `PVCache`.

cluster1.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_3

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.25 FVP_Base_Cortex-A57x2-A53x4

List of instances in FVP_Base_Cortex-A57x2-A53x4.

FVP_Base_Cortex-A57x2-A53x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.coresight_modifier

Type: `PVBusMapper`.

bp.dmc

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc.bus_slave

Type: `PVBusSlave`.

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc_phy.bus_slave

Type: `PVBusSlave`.

bp.dram_limiter

Type: `PVBusMapper`.

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_local_dap_rom.bus_slave

Type: `PVBusSlave`.

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_ram.bus_slave

Type: `PVBusSlave`.

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_usb.bus_slave

Type: `PVBusSlave`.

bp.exclusive_monitor

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

bp.exclusive_monitor.bus_mapper

Type: `PVBusMapper`.

bp.fixed_security_map

Type: `PVBusMapper`.

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash0.map

Type: `PVBusMapper`.

bp.flash0.mbs

Type: `PVBusSlave`.

bp.flash0.rmbs

Type: `PVBusSlave`.

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash1.map

Type: `PVBusMapper`.

bp.flash1.mbs

Type: `PVBusSlave`.

bp.flash1.rmbs

Type: `PVBusSlave`.

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.generic_watchdog

ARM Generic Watchdog.

Type: `MemoryMappedGenericWatchdog`.

bp.generic_watchdog.busctlbase

Type: `PVBusSlave`.

bp.generic_watchdog.busrefbase

Type: `PVBusSlave`.

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCD`.

bp.hdlcd0.busmaster

Type: `PVBusMaster`.

bp.hdlcd0.busslave

Type: `PVBusSlave`.

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slaveType: [PVBusSlave](#).**bp.pl011_uart0**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart0.busslave**Type: [PVBusSlave](#).**bp.pl011_uart0.clk_divider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.pl011_uart0.timer**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**bp.pl011_uart0.timer.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**bp.pl011_uart0.timer.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**bp.pl011_uart0.timer.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**bp.pl011_uart1**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart1.busslave**Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl1111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl1111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBusSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: filter0.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: filter1.

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: filter2.

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: filter3.

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [VE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu1
ARM Cortex-A57 CT model.
Type: `ARM_Cortex-A57`.

cluster0.cpu1.UTLB
Type: `TLB`.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: `Tlbcadi`.

cluster0.cpu1.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu1.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

cluster1

ARM Cortex-A53 Cluster CT model.

Type: `Cluster_ARM_Cortex-A53`.**cluster1.AMU**Type: `PVBusLogger`.**cluster1.AMU.mapper**Type: `PVBusMapper`.**cluster1.DAP**Type: `PVBusLogger`.**cluster1.DAP.mapper**Type: `PVBusMapper`.**cluster1.DSU**Type: `DSU`.**cluster1.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster1.MMAP**Type: `PVBusLogger`.**cluster1.MMAP.mapper**Type: `PVBusMapper`.**cluster1.acp_mapper**Type: `PVBusMapper`.**cluster1.cpu0**

ARM Cortex-A53 CT model.

Type: `ARM_Cortex-A53`.**cluster1.cpu0.UTLB**Type: `TLB`.**cluster1.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster1.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster1.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster1.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu0.l1icache**

PV Cache.

Type: `PVCache`.

cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu1

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu2

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster1.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu3
ARM Cortex-A53 CT model.
Type: `ARM_Cortex-A53`.

cluster1.cpu3.UTLB
Type: `TLB`.

cluster1.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster1.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster1.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.ext_bus
Type: `PVBusLogger`.

cluster1.ext_bus.mapper
Type: `PVBusMapper`.

cluster1.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster1.l2_cache
PV Cache.
Type: `PVCache`.

cluster1.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster1.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster1_labeller
Type: [Labeller](#).

cluster1_labeller.pvbusmodifier
Type: [PVBUSMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBUSLogger](#).

dapmemlogger.mapperType: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.26 FVP_Base_Cortex-A57x4

List of instances in FVP_Base_Cortex-A57x4.

FVP_Base_Cortex-A57x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_0_1.busslave

Type: `PVBusSlave`.

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_2_3.busslave

Type: `PVBusSlave`.

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map
Type: [PVBusMapper](#).

bp.flash1.mbs
Type: [PVBusSlave](#).

bp.flash1.rmbs
Type: [PVBusSlave](#).

bp.flashloader0
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.flashloader1
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.generic_watchdog
ARM Generic Watchdog.
Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase
Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmb

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave
Type: [PVBusSlave](#).

bp.refcounter
Memory Mapped Counter Module for Generic Timers.
Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]
Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]
Type: [PVBusSlave](#).

bp.reset_or
Or Gate.
Type: [OrGate](#).

bp.rl_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smisc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`bp.trusted_watchdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`bp.trusted_watchdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`bp.tzc_400`

TrustZone Address Space Controller.

Type: [TZC_400](#).

`bp.tzc_400.apbslave[0]`

Type: [PVBusSlave](#).

`bp.tzc_400.filter0`

Type: `filter0`.

`bp.tzc_400.filter0.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter1`

Type: `filter1`.

`bp.tzc_400.filter1.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter2`

Type: `filter2`.

`bp.tzc_400.filter2.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter3`

Type: `filter3`.

`bp.tzc_400.filter3.BusMapper`

Type: [PVBusMapper](#).

`bp.utility_bus_map0`

Type: [PVBusMapper](#).

`bp.utility_bus_map1`

Type: [PVBusMapper](#).

`bp.utility_bus_map2`

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.

cluster0.cpu1.UTLB

Type: `TLB`.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.**cluster0.cpu3.UTLB**Type: `TLB`.**cluster0.cpu3.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu3.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.ext_bus**Type: `PVBusLogger`.**cluster0.ext_bus.mapper**Type: `PVBusMapper`.**cluster0.gic_cpuif_decoder_cluster**Type: `GICv3CPUInterfaceDecoder`.**cluster0.l2_cache**

PV Cache.

Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifierType: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_tl**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#)

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.27 FVP_Base_Cortex-A57x4-A35x4

List of instances in `FVP_Base_Cortex-A57x4-A35x4`.

FVP_Base_Cortex-A57x4-A35x4 instances

address_map_terminator

Type: `PVBusMapper`.

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmb

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`bp.trusted_watchdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`bp.trusted_watchdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`bp.tzc_400`

TrustZone Address Space Controller.

Type: [TZC_400](#).

`bp.tzc_400.apbslave[0]`

Type: [PVBusSlave](#).

`bp.tzc_400.filter0`

Type: `filter0`.

`bp.tzc_400.filter0.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter1`

Type: `filter1`.

`bp.tzc_400.filter1.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter2`

Type: `filter2`.

`bp.tzc_400.filter2.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter3`

Type: `filter3`.

`bp.tzc_400.filter3.BusMapper`

Type: [PVBusMapper](#).

`bp.utility_bus_map0`

Type: [PVBusMapper](#).

`bp.utility_bus_map1`

Type: [PVBusMapper](#).

`bp.utility_bus_map2`

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A57 Cluster CT model.

Type: [Cluster_ARM_Cortex-A57](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu0.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.

cluster0.cpu1.UTLB

Type: `TLB`.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu2.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu2.l1dcache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu2.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu2.l1icache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu2.l1icache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu3**

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).**cluster0.cpu3.UTLB**Type: [TLB](#).**cluster0.cpu3.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu3.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu3.l1dcache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu3.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu3.l1icache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu3.l1icache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.ext_bus**Type: [PVBUSLogger](#).**cluster0.ext_bus.mapper**Type: [PVBUSMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]
Type: PVBusSlave.

cluster0.l2_cache.upstream[2]
Type: PVBusSlave.

cluster0.l2_cache.upstream[3]
Type: PVBusSlave.

cluster0.l2_cache.upstream[4]
Type: PVBusSlave.

cluster0.l2_cache.upstream[5]
Type: PVBusSlave.

cluster0.l2_cache.upstream[6]
Type: PVBusSlave.

cluster0.l2_cache.upstream[7]
Type: PVBusSlave.

cluster0.l2_cache.upstream[8]
Type: PVBusSlave.

cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

cluster0.l2_flusher
Type: AsyncCacheFlushUnit.

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

cluster1
ARM Cortex-A35 Cluster CT model.
Type: Cluster_ARM_Cortex-A35.

cluster1.AMU
Type: [PVBusLogger](#).

cluster1.AMU.mapper
Type: [PVBusMapper](#).

cluster1.DAP
Type: [PVBusLogger](#).

cluster1.DAP.mapper
Type: [PVBusMapper](#).

cluster1.DSU
Type: dsu.

cluster1.DSU.mpam_busslave
Type: [PVBusSlave](#).

cluster1.MMAP
Type: [PVBusLogger](#).

cluster1.MMAP.mapper
Type: [PVBusMapper](#).

cluster1.acp_mapper
Type: [PVBusMapper](#).

cluster1.cpu0
ARM Cortex-A35 CT model.
Type: [ARM_Cortex-A35](#).

cluster1.cpu0.UTLB
Type: TLB.

cluster1.cpu0.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: pvcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: pvcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: pvcache.

cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1.l1icache

PV Cache.

Type: pvcache.

cluster1.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2

ARM Cortex-A35 CT model.

Type: ARM_Cortex-A35.

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu2.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu2.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.cpu3

ARM Cortex-A35 CT model.

Type: [ARM_Cortex-A35](#).

cluster1.cpu3.UTLB

Type: [TLB](#).

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu3.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu3.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.ext_bus

Type: [PVBUSLogger](#).

cluster1.ext_bus.mapper

Type: [PVBUSMapper](#).

cluster1.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster1.l2_cache

PV Cache.

Type: `PVCache`.

cluster1.l2_cache.upstream[0]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[10]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[11]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[12]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[13]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[14]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[15]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[16]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[1]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[2]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[3]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[4]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[5]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[6]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[7]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[8]

Type: `PVBusSlave`.

cluster1.l2_cache.upstream[9]

Type: `PVBusSlave`.

cluster1.l2_flusher

Type: AsyncCacheFlushUnit.

cluster1_labeller

Type: Labeller.

cluster1_labeller.pvbusmodifier

Type: PVBusMapper.

dapmemlogger

Bus Logger.

Type: PVBusLogger.

dapmemlogger.mapper

Type: PVBusMapper.

elfloader

ELF loader component.

Type: ElfLoader.

elfloader.pvbus_busmaster

Type: PVBusMaster.

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: GIC_IRI.

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_3

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_3

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).

pctl.utility_bus2Type: [PVBusMaster](#).**pctl.utility_bus3**Type: [PVBusMaster](#).

12.28 FVP_Base_Cortex-A57x4-A53x4

List of instances in FVP_Base_Cortex-A57x4-A53x4.

FVP_Base_Cortex-A57x4-A53x4 instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiterType: [PVBusMapper](#).**bp.dummy_local_dap_rom**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_local_dap_rom.bus_slave**Type: [PVBusSlave](#).**bp.dummy_ram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_ram.bus_slave**Type: [PVBusSlave](#).**bp.dummy_usb**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_usb.bus_slave**Type: [PVBusSlave](#).**bp.exclusive_monitor**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).**bp.exclusive_monitor.bus_mapper**Type: [PVBusMapper](#).**bp.fixed_security_map**Type: [PVBusMapper](#).**bp.flash0**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash0.map**Type: [PVBusMapper](#).**bp.flash0.mbs**Type: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1
Type: filter1.

bp.tzc_400.filter1.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.

Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.vis.recorder.playbackTimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.vis.recorder.playbackTimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.vis.recorder.playbackTimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.vis.recorder.recordingDivider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.vram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.vram.bus_slave`

Type: `PVBusSlave`.

`cci400`

Cache Coherent Interconnect for AXI4 ACE.

Type: `CCI400`.

`cci400.cciinterconnect`

Type: `PVCache`.

`cci400.cciregisters`

Programmer-visible memory mapped registers for use by CCI400.

Type: `CCIRegisters`.

`cci400.cciregisters.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

cci400.cciregisters.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

cci400.cciregisters.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

cci400.cciregisters.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

clockdivider0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

clockdivider1

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

cluster0

ARM Cortex-A57 Cluster CT model.

Type: `Cluster_ARM_Cortex-A57`.

cluster0.AMU

Type: `PVBusLogger`.

cluster0.AMU.mapper

Type: `PVBusMapper`.

cluster0.DAP

Type: `PVBusLogger`.

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: dsu.

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A57 CT model.

Type: [ARM_Cortex-A57](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A57 CT model.

Type: `ARM_Cortex-A57`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster0.cpu3.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu3.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu3.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu3.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu3.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.ext_bus
Type: pVBusLogger.

cluster0.ext_bus.mapper
Type: pVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: pVCache.

cluster0.l2_cache.upstream[0]
Type: pVBusSlave.

cluster0.l2_cache.upstream[10]
Type: pVBusSlave.

cluster0.l2_cache.upstream[11]
Type: pVBusSlave.

cluster0.l2_cache.upstream[12]
Type: pVBusSlave.

cluster0.l2_cache.upstream[13]
Type: pVBusSlave.

cluster0.l2_cache.upstream[14]
Type: pVBusSlave.

cluster0.l2_cache.upstream[15]
Type: pVBusSlave.

cluster0.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

cluster1

ARM Cortex-A53 Cluster CT model.

Type: [cluster_ARM_Cortex-A53](#).

cluster1.AMU

Type: [PVBUSLogger](#).

cluster1.AMU.mapper

Type: [PVBUSMapper](#).

cluster1.DAP

Type: [PVBUSLogger](#).

cluster1.DAP.mapper

Type: [PVBUSMapper](#).

cluster1.DSU

Type: DSU.

cluster1.DSU.mpam_busslave

Type: PVBusSlave.

cluster1.MMAP

Type: PVBusLogger.

cluster1.MMAP.mapper

Type: PVBusMapper.

cluster1.acp_mapper

Type: PVBusMapper.

cluster1.cpu0

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu2.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu3

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu3.UTLB

Type: TLB.

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster1.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.ext_bus
Type: `PVBusLogger`.

cluster1.ext_bus.mapper
Type: `PVBusMapper`.

cluster1.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster1.l2_cache
PV Cache.
Type: `PVCache`.

cluster1.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_3

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_3

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.29 FVP_Base_Cortex-A65

List of instances in `FVP_Base_Cortex-A65`.

FVP_Base_Cortex-A65 instances

address_map_terminator

Type: `PVBusMapper`.

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_0_1.busslave

Type: `PVBusSlave`.

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A65 Cluster CT model.

Type: `Cluster_ARM_Cortex-A65`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.l3_flusher**Type: `AsyncCacheFlushUnit`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[10]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[11]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[12]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[13]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[14]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[15]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[16]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[17]**Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[1]Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[2]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[3]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[4]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[5]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[6]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[7]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[8]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[9]**Type: [PVBUSSlave](#).**cluster0.MMAP**Type: [PVBUSLogger](#).**cluster0.MMAP.mapper**Type: [PVBUSMapper](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.thread0**

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).**cluster0.cpu0.thread0.UTLB**Type: [TLB](#).**cluster0.cpu0.thread0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.thread0.l1dcache**Type: [PVCache](#).**cluster0.cpu0.thread0.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu0.thread0.l1icache**Type: [PVCache](#).

cluster0.cpu0.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l2cache

Type: [PVCACHE](#).

cluster0.cpu0.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu0.thread1.UTLB

Type: [TLB](#).

cluster0.cpu0.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TLBCADI](#).

cluster0.cpu1.thread0

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu1.thread0.UTLB

Type: [TLB](#).

cluster0.cpu1.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.thread0.l1dcache

Type: [PVCACHE](#).

cluster0.cpu1.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread0.l1icache

Type: [PVCACHE](#).

cluster0.cpu1.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread0.l2cache

Type: [PVCACHE](#).

cluster0.cpu1.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu1.thread1.UTLB

Type: TLB.

cluster0.cpu1.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.thread0

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu2.thread0.UTLB

Type: TLB.

cluster0.cpu2.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.thread0.l1dcache

Type: PVBUSSlave.

cluster0.cpu2.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.thread0.l1icache

Type: PVBUSSlave.

cluster0.cpu2.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.thread0.l2cache

Type: PVBUSSlave.

cluster0.cpu2.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu2.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu2.thread1.UTLB

Type: TLB.

cluster0.cpu2.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu3.thread0

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu3.thread0.UTLB

Type: TLB.

cluster0.cpu3.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.thread0.l1dcache

Type: PVCache.

cluster0.cpu3.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l1icache

Type: PVCache.

cluster0.cpu3.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l2cache

Type: PVCache.

cluster0.cpu3.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu3.thread1.UTLB

Type: TLB.

cluster0.cpu3.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu4.thread0

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu4.thread0.UTLB

Type: TLB.

cluster0.cpu4.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.thread0.l1dcache

Type: PVPCache.

cluster0.cpu4.thread0.l1dcache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu4.thread0.l1icache

Type: PVPCache.

cluster0.cpu4.thread0.l1icache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu4.thread0.l2cache

Type: PVPCache.

cluster0.cpu4.thread0.l2cache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu4.thread0.l2cache.upstream[1]

Type: PVPBusSlave.

cluster0.cpu4.thread1

ARM Cortex-A65 CT model.

Type: ARM_Cortex-A65.

cluster0.cpu4.thread1.UTLB

Type: TLB.

cluster0.cpu4.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.thread0

ARM Cortex-A65 CT model.

Type: ARM_Cortex-A65.

cluster0.cpu5.thread0.UTLB

Type: TLB.

cluster0.cpu5.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.thread0.l1dcache

Type: PVPCache.

cluster0.cpu5.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread0.l1icache

Type: [PVCACHE](#).

cluster0.cpu5.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread0.l2cache

Type: [PVCACHE](#).

cluster0.cpu5.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu5.thread1.UTLB

Type: [TLB](#).

cluster0.cpu5.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: [TLBCADI](#).

cluster0.cpu6.thread0

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu6.thread0.UTLB

Type: [TLB](#).

cluster0.cpu6.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu6.thread0.l1dcache

Type: [PVCACHE](#).

cluster0.cpu6.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.thread0.l1icache

Type: [PVCACHE](#).

cluster0.cpu6.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu6.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu6.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu6.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu6.thread1.UTLB

Type: [TLB](#).

cluster0.cpu6.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu7.thread0

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).

cluster0.cpu7.thread0.UTLB

Type: [TLB](#).

cluster0.cpu7.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu7.thread0.l1dcache

Type: [PVCache](#).

cluster0.cpu7.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu7.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu7.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu7.thread1

ARM Cortex-A65 CT model.

Type: [ARM_Cortex-A65](#).**cluster0.cpu7.thread1.UTLB**

Type: TLB.

cluster0.cpu7.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.ext_busType: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**

Type: GICv3CPUInterfaceDecoder.

cluster0_labellerType: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7_1

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).

pctl.utility_bus2Type: [PVBusMaster](#).**pctl.utility_bus3**Type: [PVBusMaster](#).

12.30 FVP_Base_Cortex-A65AE

List of instances in FVP_Base_Cortex-A65AE.

FVP_Base_Cortex-A65AE instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiterType: [PVBusMapper](#).**bp.dummy_local_dap_rom**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_local_dap_rom.bus_slave**Type: [PVBusSlave](#).**bp.dummy_ram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_ram.bus_slave**Type: [PVBusSlave](#).**bp.dummy_usb**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_usb.bus_slave**Type: [PVBusSlave](#).**bp.exclusive_monitor**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).**bp.exclusive_monitor.bus_mapper**Type: [PVBusMapper](#).**bp.fixed_security_map**Type: [PVBusMapper](#).**bp.flash0**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash0.map**Type: [PVBusMapper](#).**bp.flash0.mbs**Type: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hd1cd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbuslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: filter0.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1
Type: filter1.

bp.tzc_400.filter1.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.

Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.vis.recorder.playbackTimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.vis.recorder.playbackTimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.vis.recorder.playbackTimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.vis.recorder.recordingDivider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.vram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.vram.bus_slave`

Type: `PVBusSlave`.

`cci400`

Cache Coherent Interconnect for AXI4 ACE.

Type: `CCI400`.

`cci400.cciinterconnect`

Type: `PVCache`.

`cci400.cciregisters`

Programmer-visible memory mapped registers for use by CCI400.

Type: `CCIRegisters`.

`cci400.cciregisters.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

cci400.cciregisters.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

cci400.cciregisters.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

cci400.cciregisters.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

clockdivider0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

cluster0

ARM Cortex-A65AE Cluster CT model.

Type: `Cluster_ARM_Cortex-A65AE`.

cluster0.AMU

Type: `PVBusLogger`.

cluster0.AMU.mapper

Type: `PVBusMapper`.

cluster0.DAP

Type: `PVBusLogger`.

cluster0.DAP.mapper

Type: `PVBusMapper`.

cluster0.DSU

Type: `DSU`.

cluster0.DSU.13_flusher

Type: `AsyncCacheFlushUnit`.

cluster0.DSU.mpam_busslave

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCACHE](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[10]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[11]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[12]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.MMAP

Type: [PVBUSLogger](#).

cluster0.MMAP.mapper

Type: [PVBUSMapper](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu0.thread0.UTLB

Type: [TLB](#).

cluster0.cpu0.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.thread0.l1dcache

Type: [PVCACHE](#).

cluster0.cpu0.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l1icache

Type: [PVCACHE](#).

cluster0.cpu0.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l2cache

Type: [PVCACHE](#).

cluster0.cpu0.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread1

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu0.thread1.UTLB

Type: [TLB](#).

cluster0.cpu0.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu1.thread0.UTLB

Type: TLB.

cluster0.cpu1.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.thread0.l1dcache

Type: PVCache.

cluster0.cpu1.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.thread0.l1icache

Type: PVCache.

cluster0.cpu1.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.thread0.l2cache

Type: PVCache.

cluster0.cpu1.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1.thread1

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu1.thread1.UTLB

Type: TLB.

cluster0.cpu1.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu2.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu2.thread0.UTLB

Type: TLB.

cluster0.cpu2.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.thread0.l1dcache

Type: PVPCache.

cluster0.cpu2.thread0.l1dcache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu2.thread0.l1icache

Type: PVPCache.

cluster0.cpu2.thread0.l1icache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu2.thread0.l2cache

Type: PVPCache.

cluster0.cpu2.thread0.l2cache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu2.thread0.l2cache.upstream[1]

Type: PVPBusSlave.

cluster0.cpu2.thread1

ARM Cortex-A65AE CT model.

Type: ARM_Cortex-A65AE.

cluster0.cpu2.thread1.UTLB

Type: TLB.

cluster0.cpu2.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu3.thread0

ARM Cortex-A65AE CT model.

Type: ARM_Cortex-A65AE.

cluster0.cpu3.thread0.UTLB

Type: TLB.

cluster0.cpu3.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.thread0.l1dcache

Type: PVPCache.

cluster0.cpu3.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu3.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu3.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3.thread1

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu3.thread1.UTLB

Type: [TLB](#).

cluster0.cpu3.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu4.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu4.thread0.UTLB

Type: [TLB](#).

cluster0.cpu4.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.thread0.l1dcache

Type: [PVCache](#).

cluster0.cpu4.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu4.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu4.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4.thread1

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu4.thread1.UTLB

Type: [TLB](#).

cluster0.cpu4.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu5.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.cpu5.thread0.UTLB

Type: [TLB](#).

cluster0.cpu5.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu5.thread0.l1dcache

Type: [PVCache](#).

cluster0.cpu5.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu5.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu5.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5.thread1

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).**cluster0.cpu5.thread1.UTLB**

Type: TLB.

cluster0.cpu5.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu6.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).**cluster0.cpu6.thread0.UTLB**

Type: TLB.

cluster0.cpu6.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.thread0.l1dcache

Type: PVCache.

cluster0.cpu6.thread0.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu6.thread0.l1icache**

Type: PVCache.

cluster0.cpu6.thread0.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu6.thread0.l2cache**

Type: PVCache.

cluster0.cpu6.thread0.l2cache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu6.thread0.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu6.thread1**

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).**cluster0.cpu6.thread1.UTLB**

Type: TLB.

cluster0.cpu6.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu7.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).**cluster0.cpu7.thread0.UTLB**

Type: TLB.

cluster0.cpu7.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu7.thread0.l1dcache

Type: Pvcache.

cluster0.cpu7.thread0.l1dcache.upstream[0]Type: [PvBusSlave](#).**cluster0.cpu7.thread0.l1icache**

Type: Pvcache.

cluster0.cpu7.thread0.l1icache.upstream[0]Type: [PvBusSlave](#).**cluster0.cpu7.thread0.l2cache**

Type: Pvcache.

cluster0.cpu7.thread0.l2cache.upstream[0]Type: [PvBusSlave](#).**cluster0.cpu7.thread0.l2cache.upstream[1]**Type: [PvBusSlave](#).**cluster0.cpu7.thread1**

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).**cluster0.cpu7.thread1.UTLB**

Type: TLB.

cluster0.cpu7.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.ext_busType: [PvBusLogger](#).**cluster0.ext_bus.mapper**Type: [PvBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**

Type: GICv3CPUInterfaceDecoder.

cluster0_labellerType: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_2**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_2_0**Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7_1
Type: GICv3Redistributor.

gic_distributor.rd_tl
Type: GICv3Distributor.

pctl
Base Platforms Power Controller.
Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.31 FVP_Base_Cortex-A65AE+Cortex-A76AE

List of instances in FVP_Base_Cortex-A65AE+Cortex-A76AE.

FVP_Base_Cortex-A65AE+Cortex-A76AE instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A65AE_Cortex-A76AE Cluster CT model.

Type: [Cluster_ARM_Cortex-A65AE_Cortex-A76AE](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.ext_bus**Type: `PVBusLogger`.**cluster0.ext_bus.mapper**Type: `PVBusMapper`.**cluster0.gic_cpuif_decoder_cluster**Type: `GICv3CPUInterfaceDecoder`.**cluster0.subcluster0**

ARM Cortex-A65AE Cluster CT model.

Type: `Subcluster_ARM_Cortex-A65AE`.**cluster0.subcluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.subcluster0.cpu0.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.subcluster0.cpu0.thread0.UTLB

Type: TLB.

cluster0.subcluster0.cpu0.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster0.cpu0.thread0.l1dcache

Type: Pvcache.

cluster0.subcluster0.cpu0.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.subcluster0.cpu0.thread0.l1icache

Type: Pvcache.

cluster0.subcluster0.cpu0.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.subcluster0.cpu0.thread0.l2cache

Type: Pvcache.

cluster0.subcluster0.cpu0.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.subcluster0.cpu0.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.subcluster0.cpu0.thread1

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.subcluster0.cpu0.thread1.UTLB

Type: TLB.

cluster0.subcluster0.cpu0.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.subcluster0.cpu1.thread0

ARM Cortex-A65AE CT model.

Type: [ARM_Cortex-A65AE](#).

cluster0.subcluster0.cpu1.thread0.UTLB

Type: TLB.

cluster0.subcluster0.cpu1.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster0.cpu1.thread0.l1dcache

Type: PVCache.

cluster0.subcluster0.cpu1.thread0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster0.cpu1.thread0.l1icache

Type: PVCache.

cluster0.subcluster0.cpu1.thread0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster0.cpu1.thread0.l2cache

Type: PVCache.

cluster0.subcluster0.cpu1.thread0.l2cache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster0.cpu1.thread0.l2cache.upstream[1]

Type: PVBusSlave.

cluster0.subcluster0.cpu1.thread1

ARM Cortex-A65AE CT model.

Type: ARM_Cortex-A65AE.

cluster0.subcluster0.cpu1.thread1.UTLB

Type: TLB.

cluster0.subcluster0.cpu1.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster1

ARM Cortex-A76AE Cluster CT model.

Type: Subcluster_ARM_Cortex-A76AE.

cluster0.subcluster1.cpu0

ARM Cortex-A76AE CT model.

Type: ARM_Cortex-A76AE.

cluster0.subcluster1.cpu0.UTLB

Type: TLB.

cluster0.subcluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.subcluster1.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster0.subcluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.subcluster1.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster0.subcluster1.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu0.l2cache

PV Cache.

Type: `PVCache`.

cluster0.subcluster1.cpu0.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu0.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu1

ARM Cortex-A76AE CT model.

Type: `ARM_Cortex-A76AE`.

cluster0.subcluster1.cpu1.UTLB

Type: `TLB`.

cluster0.subcluster1.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.subcluster1.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.subcluster1.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.subcluster1.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu1.l2cache

PV Cache.

Type: `PVCache`.

cluster0.subcluster1.cpu1.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu1.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.subcluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.subcluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0_labeller**Type: `Labeller`.**cluster0_labeller.pvbusmodifier**Type: `PVBusMapper`.**dapmemlogger**

Bus Logger.

Type: `PVBusLogger`.**dapmemlogger.mapper**Type: `PVBusMapper`.**elfloader**

ELF loader component.

Type: `ElfLoader`.**elfloader.pvbus_busmaster**Type: `PVBusMaster`.**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: `GIC_IRI`.**gic_distributor.rd_0**Type: `GICv3RedistributorInternal`.**gic_distributor.rd_0_0**Type: `GICv3RedistributorInternal`.**gic_distributor.rd_0_0_0**Type: `GICv3RedistributorInternal`.**gic_distributor.rd_0_0_0_0**Type: `GICv3Redistributor`.**gic_distributor.rd_0_0_0_1**Type: `GICv3Redistributor`.**gic_distributor.rd_0_0_1**Type: `GICv3RedistributorInternal`.**gic_distributor.rd_0_0_1_0**Type: `GICv3Redistributor`.**gic_distributor.rd_0_0_1_1**Type: `GICv3Redistributor`.**gic_distributor.rd_0_0_2**Type: `GICv3RedistributorInternal`.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_8
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_9
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0
Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: `GICv3Distributor`.

pctl

Base Platforms Power Controller.

Type: `Base_PowerController`.

pctl.busslave

Type: `PVBusSlave`.

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.32 FVP_Base_Cortex-A710

List of instances in FVP_Base_Cortex-A710.

FVP_Base_Cortex-A710 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A710 Cluster CT model.

Type: [Cluster_ARM_Cortex-A710](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUv1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBUSSlave](#).

cluster0.DSU.PPU_core0
Type: PPUv1.

cluster0.DSU.PPU_core0.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core1
Type: PPUv1.

cluster0.DSU.PPU_core1.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core10
Type: PPUv1.

cluster0.DSU.PPU_core10.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core11
Type: PPUv1.

cluster0.DSU.PPU_core11.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core2
Type: PPUv1.

cluster0.DSU.PPU_core2.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core3
Type: PPUv1.

cluster0.DSU.PPU_core3.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core4
Type: PPUv1.

cluster0.DSU.PPU_core4.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core5
Type: PPUv1.

cluster0.DSU.PPU_core5.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core6
Type: PPUv1.

cluster0.DSU.PPU_core6.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core7
Type: PPUv1.

cluster0.DSU.PPU_core7.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core8

Type: [PPUv1](#).

cluster0.DSU.PPU_core8.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core9

Type: [PPUv1](#).

cluster0.DSU.PPU_core9.busslave

Type: [PVBusSlave](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[17]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[18]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[19]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.DSU.utility_slave[0]
Type: [PVBUSSlave](#).

cluster0.MMAP
Type: [PVBUSLogger](#).

cluster0.MMAP.mapper
Type: [PVBUSMapper](#).

cluster0.cpu0
ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu1

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu10

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu10.UTLB

Type: [TLB](#).

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu10.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu10.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu11

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu11.UTLB

Type: TLB.

cluster0.cpu11.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu11.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu11.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu11.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu11.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu11.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu11.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu11.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu11.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-A710 CT model.

Type: `ARM_Cortex-A710`.**cluster0.cpu3.UTLB**Type: `TLB`.**cluster0.cpu3.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu3.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu4

ARM Cortex-A710 CT model.

Type: `ARM_Cortex-A710`.

cluster0.cpu4.UTLB

Type: `TLB`.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu4.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu4.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu5

ARM Cortex-A710 CT model.

Type: `ARM_Cortex-A710`.

cluster0.cpu5.UTLB

Type: `TLB`.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu5.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu5.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu6

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu6.UTLB

Type: [TLB](#).

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu6.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu6.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu6.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu6.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu6.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.l2cache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu6.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu7

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu7.UTLB

Type: [TLB](#).

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: [tlbcadi](#).

cluster0.cpu7.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu7.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu7.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu7.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu7.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu7.l2cache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu7.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu7.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu8

ARM Cortex-A710 CT model.

Type: [ARM_Cortex-A710](#).

cluster0.cpu8.UTLB

Type: TLB.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu8.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu8.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu8.l1dcache.upstream[0]

Type: PVBUSSlave.

cluster0.cpu8.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu8.l1icache.upstream[0]

Type: PVBUSSlave.

cluster0.cpu8.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu8.l2cache.upstream[0]

Type: PVBUSSlave.

cluster0.cpu8.l2cache.upstream[1]

Type: PVBUSSlave.

cluster0.cpu9

ARM Cortex-A710 CT model.

Type: ARM_Cortex-A710.

cluster0.cpu9.UTLB

Type: TLB.

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu9.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu9.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu9.l1dcache.upstream[0]

Type: PVBUSSlave.

cluster0.cpu9.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu9.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu9.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_11
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_11_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0
Type: [PVBusMaster](#).

pctl.utility_bus1
Type: [PVBusMaster](#).

pctl.utility_bus2
Type: [PVBusMaster](#).

pctl.utility_bus3
Type: [PVBusMaster](#).

12.33 FVP_Base_Cortex-A715

List of instances in FVP_Base_Cortex-A715.

FVP_Base_Cortex-A715 instances

address_map_terminator
Type: [PVBusMapper](#).

bp
Peripherals and address map for the Base Platform.
Type: [BasePlatformPeripherals](#).

bp.Timer_0_1
ARM Dual-Timer Module(SP804).
Type: [SP804_Timer](#).

bp.Timer_0_1.busslave
Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.counter0
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A715 Cluster CT model.

Type: [Cluster_ARM_Cortex-A715](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core10

Type: PPUv1.

cluster0.DSU.PPU_core10.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core11

Type: PPUv1.

cluster0.DSU.PPU_core11.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core6
Type: PPUv1.

cluster0.DSU.PPU_core6.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core7
Type: PPUv1.

cluster0.DSU.PPU_core7.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core8
Type: PPUv1.

cluster0.DSU.PPU_core8.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core9
Type: PPUv1.

cluster0.DSU.PPU_core9.busslave
Type: PVBusSlave.

cluster0.DSU.l3_flusher
Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave
Type: PVBusSlave.

cluster0.DSU.shared_cache
PV Cache.
Type: pVCache.

cluster0.DSU.shared_cache.upstream[0]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[10]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[11]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[12]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[13]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[14]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[15]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l2cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.cpu10
ARM Cortex-A715 CT model.
Type: [ARM_Cortex-A715](#).

cluster0.cpu10.UTLB
Type: [TLB](#).

cluster0.cpu10.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu10.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu10.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu10.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu10.l1icache
PV Cache.

Type: `PVCache`.

cluster0.cpu10.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu10.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu10.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu10.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu11
ARM Cortex-A715 CT model.
Type: `ARM_Cortex-A715`.

cluster0.cpu11.UTLB
Type: `TLB`.

cluster0.cpu11.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu11.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu11.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu2
ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu3.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu4.UTLB

Type: [TLB](#).

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu4.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu6

ARM Cortex-A715 CT model.

Type: [ARM_Cortex-A715](#).

cluster0.cpu6.UTLB

Type: TLB.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu6.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu7**

ARM Cortex-A715 CT model.

Type: `ARM_Cortex-A715`.**cluster0.cpu7.UTLB**Type: `TLB`.**cluster0.cpu7.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu7.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu7.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu8

ARM Cortex-A715 CT model.

Type: `ARM_Cortex-A715`.

cluster0.cpu8.UTLB

Type: `TLB`.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu8.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu8.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu8.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu8.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu8.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu9

ARM Cortex-A715 CT model.

Type: `ARM_Cortex-A715`.

cluster0.cpu9.UTLB

Type: `TLB`.

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu9.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu9.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_11

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_11_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_4

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_5

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.34 FVP_Base_Cortex-A720

List of instances in FVP_Base_Cortex-A720.

FVP_Base_Cortex-A720 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.coresight_modifier

Type: `PVBusMapper`.

bp.dmc

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc.bus_slave

Type: `PVBusSlave`.

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc_phy.bus_slave

Type: `PVBusSlave`.

bp.dram_limiter

Type: `PVBusMapper`.

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_local_dap_rom.bus_slave

Type: `PVBusSlave`.

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_ram.bus_slave

Type: `PVBusSlave`.

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_usb.bus_slave

Type: `PVBusSlave`.

bp.exclusive_monitor

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

bp.exclusive_monitor.bus_mapper

Type: `PVBusMapper`.

bp.fixed_security_map

Type: `PVBusMapper`.

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash0.map

Type: `PVBusMapper`.

bp.flash0.mbs

Type: `PVBusSlave`.

bp.flash0.rmbs

Type: `PVBusSlave`.

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash1.map

Type: `PVBusMapper`.

bp.flash1.mbs

Type: `PVBusSlave`.

bp.flash1.rmbs

Type: `PVBusSlave`.

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.generic_watchdog

ARM Generic Watchdog.

Type: `MemoryMappedGenericWatchdog`.

bp.generic_watchdog.busctlbase

Type: `PVBusSlave`.

bp.generic_watchdog.busrefbase

Type: `PVBusSlave`.

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCD`.

bp.hdlcd0.busmaster

Type: `PVBusMaster`.

bp.hdlcd0.busslave

Type: `PVBusSlave`.

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.hdlcd0.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.hdlcd0.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.hdlcd0.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.hdlcd0_labeller`

Type: `Labeller`.

`bp.hdlcd0_labeller.pvbusmodifier`

Type: `PVBusMapper`.

`bp.hostbridge`

Host Socket Interface Component.

Type: `HostBridge`.

`bp.lcd_security_map`

Type: `PVBusMapper`.

`bp.ls64_testing_fifo`

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

`bp.ls64_testing_fifo.pvbuslave`

Type: `PVBusSlave`.

`bp.mmc`

Generic Multimedia Card.

Type: `MMC`.

`bp.mmc.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slaveType: [PVBusSlave](#).**bp.pl011_uart0**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart0.busslave**Type: [PVBusSlave](#).**bp.pl011_uart0.clk_divider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.pl011_uart0.timer**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**bp.pl011_uart0.timer.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**bp.pl011_uart0.timer.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**bp.pl011_uart0.timer.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**bp.pl011_uart1**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart1.busslave**Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl1111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl1111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBusSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: filter0.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: filter1.

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: filter2.

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: filter3.

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [VE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A720 Cluster CT model.

Type: `Cluster_ARM_Cortex-A720`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.PPU_cluster**Type: `PPUv1`.**cluster0.DSU.PPU_cluster.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core0**Type: `PPUv1`.**cluster0.DSU.PPU_core0.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core1**Type: `PPUv1`.**cluster0.DSU.PPU_core1.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core10**Type: `PPUv1`.**cluster0.DSU.PPU_core10.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core11**Type: `PPUv1`.**cluster0.DSU.PPU_core11.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core12**Type: `PPUv1`.**cluster0.DSU.PPU_core12.busslave**Type: `PVBusSlave`.

cluster0.DSU.PPU_core13
Type: PPUv1.

cluster0.DSU.PPU_core13.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core2
Type: PPUv1.

cluster0.DSU.PPU_core2.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core3
Type: PPUv1.

cluster0.DSU.PPU_core3.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core4
Type: PPUv1.

cluster0.DSU.PPU_core4.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core5
Type: PPUv1.

cluster0.DSU.PPU_core5.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core6
Type: PPUv1.

cluster0.DSU.PPU_core6.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core7
Type: PPUv1.

cluster0.DSU.PPU_core7.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core8
Type: PPUv1.

cluster0.DSU.PPU_core8.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core9
Type: PPUv1.

cluster0.DSU.PPU_core9.busslave
Type: PVBusSlave.

cluster0.DSU.l3_flusher
Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[17]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[18]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[19]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[20]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[21]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[22]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[23]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[24]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[26]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[27]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[28]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[29]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[30]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBUSSlave](#).

cluster0.MMAP

Type: [PVBUSLogger](#).

cluster0.MMAP.mapper

Type: [PVBUSMapper](#).

cluster0.cpu0

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu1

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu10

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu10.UTLB

Type: [TLB](#).

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu10.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu10.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu11

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu11.UTLB

Type: TLB.

cluster0.cpu11.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu11.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu11.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu11.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu11.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu11.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu11.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu11.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu11.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu12

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu12.UTLB

Type: TLB.

cluster0.cpu12.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu12.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu12.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu12.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu12.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu12.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu12.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu12.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu12.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu13**

ARM Cortex-A720 CT model.

Type: `ARM_Cortex-A720`.**cluster0.cpu13.UTLB**Type: `TLB`.**cluster0.cpu13.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu13.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu13.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu13.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu13.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu13.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu13.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu13.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu13.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A720 CT model.

Type: `ARM_Cortex-A720`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A720 CT model.

Type: `ARM_Cortex-A720`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu4.UTLB

Type: [TLB](#).

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu4.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu5.UTLB

Type: [TLB](#).

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu5.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu5.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu5.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu6

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).

cluster0.cpu6.UTLB

Type: TLB.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu6.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu6.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu6.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l2cache.upstream[0]

Type: PVBusSlave.

cluster0.cpu6.l2cache.upstream[1]

Type: PVBusSlave.

cluster0.cpu7

ARM Cortex-A720 CT model.

Type: ARM_Cortex-A720.

cluster0.cpu7.UTLB

Type: TLB.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu7.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu7.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu7.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu7.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu8**

ARM Cortex-A720 CT model.

Type: `ARM_Cortex-A720`.**cluster0.cpu8.UTLB**Type: `TLB`.**cluster0.cpu8.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu8.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu8.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu8.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu8.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu8.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu8.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu8.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu8.l2cache.upstream[1]**Type: `PVBusSlave`.

cluster0.cpu9

ARM Cortex-A720 CT model.

Type: [ARM_Cortex-A720](#).**cluster0.cpu9.UTLB**

Type: TLB.

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu9.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu9.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu9.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu9.l1icache**

PV Cache.

Type: PVCache.

cluster0.cpu9.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu9.l2cache**

PV Cache.

Type: PVCache.

cluster0.cpu9.l2cache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu9.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**

Type: GICv3CPUInterfaceDecoder.

cluster0_labellerType: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_11

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_11_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_12

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_12_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_13

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_13_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_8
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_9
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0
Type: GICv3Redistributor.

gic_distributor.rd_t1
Type: GICv3Distributor.

pctl
Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.35 FVP_Base_Cortex-A725

List of instances in FVP_Base_Cortex-A725.

FVP_Base_Cortex-A725 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.pl041_aaci.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.pl041_aaci.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: `PL050_KMI`.

`bp.pl050_kmi0.busslave`

Type: `PVBusSlave`.

`bp.pl050_kmi0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: `PL050_KMI`.

`bp.pl050_kmi1.busslave`

Type: `PVBusSlave`.

`bp.pl050_kmi1.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.pl111_clcd`

ARM PrimeCell Color LCD Controller(PL111).

Type: `PL111_CLCD`.

`bp.pl111_clcd.pl11x_clcd`

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

`bp.pl111_clcd.pl11x_clcd.busmaster`

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A725 Cluster CT model.

Type: [Cluster_ARM_Cortex-A725](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUv1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBUSSlave](#).

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core10

Type: PPUv1.

cluster0.DSU.PPU_core10.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core11

Type: PPUv1.

cluster0.DSU.PPU_core11.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core12

Type: PPUv1.

cluster0.DSU.PPU_core12.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core13

Type: PPUv1.

cluster0.DSU.PPU_core13.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core6

Type: [PPUv1](#).

cluster0.DSU.PPU_core6.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core7

Type: [PPUv1](#).

cluster0.DSU.PPU_core7.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core8

Type: [PPUv1](#).

cluster0.DSU.PPU_core8.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core9

Type: [PPUv1](#).

cluster0.DSU.PPU_core9.busslave

Type: [PVBusSlave](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[26]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[27]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[28]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[29]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[30]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu10**

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.**cluster0.cpu10.UTLB**Type: `TLB`.

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu10.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu10.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu10.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu10.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu10.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu10.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu10.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu10.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu11**

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.**cluster0.cpu11.UTLB**Type: `TLB`.**cluster0.cpu11.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu11.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu11.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu11.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu11.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu11.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu11.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu11.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu11.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu12

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.

cluster0.cpu12.UTLB

Type: `TLB`.

cluster0.cpu12.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu12.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu12.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu12.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu12.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu12.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu12.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu12.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu12.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu13

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu13.UTLB

Type: TLB.

cluster0.cpu13.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu13.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu13.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu13.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu13.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu13.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu4.UTLB

Type: TLB.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu4.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu6**

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.**cluster0.cpu6.UTLB**Type: `TLB`.**cluster0.cpu6.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu6.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu6.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu6.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu6.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu7

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.

cluster0.cpu7.UTLB

Type: `TLB`.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu7.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu7.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu8

ARM Cortex-A725 CT model.

Type: `ARM_Cortex-A725`.

cluster0.cpu8.UTLB

Type: `TLB`.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu8.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu8.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu9

ARM Cortex-A725 CT model.

Type: [ARM_Cortex-A725](#).

cluster0.cpu9.UTLB

Type: [TLB](#).

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu9.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu9.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu9.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1icache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu9.l2cache
PV Cache.
Type: [PVCache](#).

cluster0.cpu9.l2cache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu9.l2cache.upstream[1]
Type: [PVBusSlave](#).

cluster0.ext_bus
Type: [PVBusLogger](#).

cluster0.ext_bus.mapper
Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBusLogger](#).

dapmemlogger.mapper
Type: [PVBusMapper](#).

elfloader
ELF loader component.
Type: [ElfLoader](#).

elfloader.pvbus_busmaster
Type: [PVBusMaster](#).

gic_distributor
GIC Interrupt Redistribution Infrastructure component.
Type: [GIC_IRI](#).

gic_distributor.rd_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_11
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_11_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_12
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_12_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_13
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_13_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.36 FVP_Base_Cortex-A72x1

List of instances in FVP_Base_Cortex-A72x1.

FVP_Base_Cortex-A72x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBusSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.virtio_rng**

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).**bp.virtio_rng.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice**

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).**bp.virtio_blockdevice.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice_labeller**Type: [Labeller](#).**bp.virtio_blockdevice_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.virtio_p9device**

virtio P9 server.

Type: [VirtioP9Device](#).**bp.virtio_p9device.mmio_slave**Type: [PVBusSlave](#).**bp.virtio_p9device.virtio_master**Type: [PVBusMaster](#).**bp.virtio_p9device_labeller**Type: [Labeller](#).**bp.virtio_p9device_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.vis**

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).**bp.vis.recorder**

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).**bp.vis.recorder.playbackDivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A72 Cluster CT model.

Type: [Cluster_ARM_Cortex-A72](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.acp_mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-A72 CT model.

Type: ARM_Cortex-A72.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: PVPcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.ext_bus

Type: PVBusLogger.

cluster0.ext_bus.mapper

Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: PVPcache.

cluster0.12_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.12_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifierType: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.37 FVP_Base_Cortex-A72x1-A53x1

List of instances in FVP_Base_Cortex-A72x1-A53x1.

FVP_Base_Cortex-A72x1-A53x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PvBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PvBusMapper](#).

bp.utility_bus_map0

Type: [PvBusMapper](#).

bp.utility_bus_map1

Type: [PvBusMapper](#).

bp.utility_bus_map2

Type: [PvBusMapper](#).

bp.utility_bus_map3

Type: [PvBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PvBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A72 Cluster CT model.

Type: [cluster_ARM_Cortex-A72](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

`cluster0.cpu0.l1dcache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.cpu0.l1icache`
PV Cache.
Type: `PVCache`.

`cluster0.cpu0.l1icache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.ext_bus`
Type: `PVBusLogger`.

`cluster0.ext_bus.mapper`
Type: `PVBusMapper`.

`cluster0.gic_cpuif_decoder_cluster`
Type: `GICv3CPUInterfaceDecoder`.

`cluster0.l2_cache`
PV Cache.
Type: `PVCache`.

`cluster0.l2_cache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[10]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[11]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[12]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[13]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[14]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[15]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[16]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[1]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[2]`
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[3]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

cluster1
ARM Cortex-A53 Cluster CT model.
Type: [Cluster_ARM_Cortex-A53](#).

cluster1.AMU
Type: [PVBusLogger](#).

cluster1.AMU.mapper
Type: [PVBusMapper](#).

cluster1.DAP
Type: [PVBusLogger](#).

cluster1.DAP.mapper
Type: [PVBusMapper](#).

cluster1.DSU
Type: [DSU](#).

cluster1.DSU.mpam_busslave
Type: [PVBusSlave](#).

cluster1.MMAP
Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.acp_mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.ext_bus

Type: [PVBusLogger](#).

cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

cluster1.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster1.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster1_labeller
Type: [Labeller](#).

cluster1_labeller.pvbusmodifier
Type: [PVBUSMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBUSLogger](#).

dapmemlogger.mapperType: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_tl**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.38 FVP_Base_Cortex-A72x2

List of instances in FVP_Base_Cortex-A72x2.

FVP_Base_Cortex-A72x2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PvBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PvBusMapper](#).

bp.utility_bus_map0

Type: [PvBusMapper](#).

bp.utility_bus_map1

Type: [PvBusMapper](#).

bp.utility_bus_map2

Type: [PvBusMapper](#).

bp.utility_bus_map3

Type: [PvBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PvBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A72 Cluster CT model.

Type: `Cluster_ARM_Cortex-A72`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.acp_mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A72 CT model.

Type: `ARM_Cortex-A72`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuiif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.12_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_0_1

Type: [GICv3Redistributor](#).

gic_distributor.rd_t1

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.39 FVP_Base_Cortex-A72x2-A53x4

List of instances in FVP_Base_Cortex-A72x2-A53x4.

FVP_Base_Cortex-A72x2-A53x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A72 Cluster CT model.

Type: [Cluster_ARM_Cortex-A72](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**

Type: dsu.

cluster0.DSU.mpam_busslaveType: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: PVCache.

cluster0.cpu0.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu1**

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).**cluster0.cpu1.UTLB**

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu1.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.ext_bus
Type: pVBusLogger.

cluster0.ext_bus.mapper
Type: pVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: pVCache.

cluster0.l2_cache.upstream[0]
Type: pVBusSlave.

cluster0.l2_cache.upstream[10]
Type: pVBusSlave.

cluster0.l2_cache.upstream[11]
Type: pVBusSlave.

cluster0.l2_cache.upstream[12]
Type: pVBusSlave.

cluster0.l2_cache.upstream[13]
Type: pVBusSlave.

cluster0.l2_cache.upstream[14]
Type: pVBusSlave.

cluster0.l2_cache.upstream[15]
Type: pVBusSlave.

cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

cluster1

ARM Cortex-A53 Cluster CT model.

Type: [cluster_ARM_Cortex-A53](#).

cluster1.AMU

Type: [PVBusLogger](#).

cluster1.AMU.mapper

Type: [PVBusMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: DSU.

cluster1.DSU.mpam_busslave

Type: PVBusSlave.

cluster1.MMAP

Type: PVBusLogger.

cluster1.MMAP.mapper

Type: PVBusMapper.

cluster1.acp_mapper

Type: PVBusMapper.

cluster1.cpu0

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: PVPcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: PVPcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu2.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2.l1icache

PV Cache.

Type: PVCache.

cluster1.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu3

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu3.UTLB

Type: TLB.

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster1.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.ext_bus
Type: `PVBusLogger`.

cluster1.ext_bus.mapper
Type: `PVBusMapper`.

cluster1.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster1.l2_cache
PV Cache.
Type: `PVCache`.

cluster1.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_3

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.40 FVP_Base_Cortex-A72x4

List of instances in FVP_Base_Cortex-A72x4.

FVP_Base_Cortex-A72x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.coresight_modifier

Type: `PVBusMapper`.

bp.dmc

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc.bus_slave

Type: `PVBusSlave`.

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc_phy.bus_slave

Type: `PVBusSlave`.

bp.dram_limiter

Type: `PVBusMapper`.

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_local_dap_rom.bus_slave

Type: `PVBusSlave`.

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_ram.bus_slave

Type: `PVBusSlave`.

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_usb.bus_slave

Type: `PVBusSlave`.

bp.exclusive_monitor

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

bp.exclusive_monitor.bus_mapper

Type: `PVBusMapper`.

bp.fixed_security_map

Type: `PVBusMapper`.

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash0.map

Type: `PVBusMapper`.

bp.flash0.mbs

Type: `PVBusSlave`.

bp.flash0.rmbs

Type: `PVBusSlave`.

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash1.map

Type: `PVBusMapper`.

bp.flash1.mbs

Type: `PVBusSlave`.

bp.flash1.rmbs

Type: `PVBusSlave`.

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.generic_watchdog

ARM Generic Watchdog.

Type: `MemoryMappedGenericWatchdog`.

bp.generic_watchdog.busctlbase

Type: `PVBusSlave`.

bp.generic_watchdog.busrefbase

Type: `PVBusSlave`.

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCD`.

bp.hdlcd0.busmaster

Type: `PVBusMaster`.

bp.hdlcd0.busslave

Type: `PVBusSlave`.

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbuslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).**bp.rl_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.rl_dram.bus_slave**Type: [PVBusSlave](#).**bp.rt_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.rt_dram.bus_slave**Type: [PVBusSlave](#).**bp.s_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.s_dram.bus_slave**Type: [PVBusSlave](#).**bp.secureDRAM**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.secureDRAM.bus_slave**Type: [PVBusSlave](#).**bp.secureSRAM**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.secureSRAM.bus_slave**Type: [PVBusSlave](#).**bp.secureSRAM_limiter**Type: [PVBusMapper](#).**bp.secureflash**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.secureflash.map**Type: [PVBusMapper](#).**bp.secureflash.mbs**Type: [PVBusSlave](#).**bp.secureflash.rmbs**Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBusSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: filter0.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: filter1.

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: filter2.

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: filter3.

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [VE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A72 Cluster CT model.

Type: `Cluster_ARM_Cortex-A72`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.acp_mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A72 CT model.

Type: `ARM_Cortex-A72`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3
ARM Cortex-A72 CT model.
Type: `ARM_Cortex-A72`.

cluster0.cpu3.UTLB
Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBUSMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBUSLogger](#).

dapmemlogger.mapperType: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.41 FVP_Base_Cortex-A72x4-A53x4

List of instances in FVP_Base_Cortex-A72x4-A53x4.

FVP_Base_Cortex-A72x4-A53x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A72 Cluster CT model.

Type: [cluster_ARM_Cortex-A72](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A72 CT model.

Type: [ARM_Cortex-A72](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1

ARM Cortex-A72 CT model.

Type: `ARM_Cortex-A72`.

cluster0.cpu1.UTLB

Type: `TLB`.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A72 CT model.

Type: `ARM_Cortex-A72`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-A72 CT model.

Type: `ARM_Cortex-A72`.**cluster0.cpu3.UTLB**Type: `TLB`.**cluster0.cpu3.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu3.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.ext_bus**Type: `PVBusLogger`.**cluster0.ext_bus.mapper**Type: `PVBusMapper`.**cluster0.gic_cpuif_decoder_cluster**Type: `GICv3CPUInterfaceDecoder`.**cluster0.l2_cache**

PV Cache.

Type: `PVCache`.

cluster0.12_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.12_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

cluster1

ARM Cortex-A53 Cluster CT model.

Type: [Cluster_ARM_Cortex-A53](#).

cluster1.AMU

Type: [PVBusLogger](#).

cluster1.AMU.mapper

Type: [PVBusMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: [DSU](#).

cluster1.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.acp_mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu0.UTLB

Type: [TLB](#).

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l1icache

PV Cache.

Type: `PVCache`.**cluster1.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu1**

ARM Cortex-A53 CT model.

Type: `ARM_Cortex-A53`.**cluster1.cpu1.UTLB**Type: `TLB`.**cluster1.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster1.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster1.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster1.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster1.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu2**

ARM Cortex-A53 CT model.

Type: `ARM_Cortex-A53`.**cluster1.cpu2.UTLB**Type: `TLB`.**cluster1.cpu2.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster1.cpu2.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster1.cpu2.l1dcache**

PV Cache.

Type: `PVCache`.

cluster1.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu3

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu3.UTLB

Type: [TLB](#).

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster1.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.ext_bus

Type: [PVBusLogger](#).

cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

cluster1.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[10]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[11]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[12]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_3**Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: `GICv3Distributor`.

pctl

Base Platforms Power Controller.

Type: `Base_PowerController`.

pctl.busslave

Type: `PVBusSlave`.

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.42 FVP_Base_Cortex-A73x1

List of instances in FVP_Base_Cortex-A73x1.

FVP_Base_Cortex-A73x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A73 Cluster CT model.

Type: [Cluster_ARM_Cortex-A73](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[12]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.12_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.12_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.43 FVP_Base_Cortex-A73x1-A53x1

List of instances in FVP_Base_Cortex-A73x1-A53x1.

FVP_Base_Cortex-A73x1-A53x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10Type: [PVBusSlave](#).**bp.ap_refclk.busbase11**Type: [PVBusSlave](#).**bp.ap_refclk.busbase12**Type: [PVBusSlave](#).**bp.ap_refclk.busbase13**Type: [PVBusSlave](#).**bp.ap_refclk.busbase14**Type: [PVBusSlave](#).**bp.ap_refclk.busbase15**Type: [PVBusSlave](#).**bp.ap_refclk.busbase2**Type: [PVBusSlave](#).**bp.ap_refclk.busbase3**Type: [PVBusSlave](#).**bp.ap_refclk.busbase4**Type: [PVBusSlave](#).**bp.ap_refclk.busbase5**Type: [PVBusSlave](#).**bp.ap_refclk.busbase6**Type: [PVBusSlave](#).**bp.ap_refclk.busbase7**Type: [PVBusSlave](#).**bp.ap_refclk.busbase8**Type: [PVBusSlave](#).**bp.ap_refclk.busbase9**Type: [PVBusSlave](#).**bp.ap_refclk.busctlbase**Type: [PVBusSlave](#).**bp.audioout**

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).**bp.clock100Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A73 Cluster CT model.

Type: [Cluster_ARM_Cortex-A73](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]
Type: PVBusSlave.

cluster0.l2_cache.upstream[2]
Type: PVBusSlave.

cluster0.l2_cache.upstream[3]
Type: PVBusSlave.

cluster0.l2_cache.upstream[4]
Type: PVBusSlave.

cluster0.l2_cache.upstream[5]
Type: PVBusSlave.

cluster0.l2_cache.upstream[6]
Type: PVBusSlave.

cluster0.l2_cache.upstream[7]
Type: PVBusSlave.

cluster0.l2_cache.upstream[8]
Type: PVBusSlave.

cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

cluster0.l2_flusher
Type: AsyncCacheFlushUnit.

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

cluster1
ARM Cortex-A53 Cluster CT model.
Type: [Cluster_ARM_Cortex-A53](#).

cluster1.AMU
Type: [PVBusLogger](#).

cluster1.AMU.mapper
Type: [PVBusMapper](#).

cluster1.DAP
Type: [PVBusLogger](#).

cluster1.DAP.mapper
Type: [PVBusMapper](#).

cluster1.DSU
Type: dsu.

cluster1.DSU.mpam_busslave
Type: [PVBusSlave](#).

cluster1.MMAP
Type: [PVBusLogger](#).

cluster1.MMAP.mapper
Type: [PVBusMapper](#).

cluster1.acp_mapper
Type: [PVBusMapper](#).

cluster1.cpu0
ARM Cortex-A53 CT model.
Type: [ARM_Cortex-A53](#).

cluster1.cpu0.UTLB
Type: TLB.

cluster1.cpu0.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster1.cpu0.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache
PV Cache.
Type: PVCache.

cluster1.cpu0.l1dcache.upstream[0]
Type: PVBusSlave.

cluster1.cpu0.l1icache
PV Cache.
Type: PVCache.

cluster1.cpu0.l1icache.upstream[0]
Type: PVBusSlave.

cluster1.ext_bus
Type: PVBusLogger.

cluster1.ext_bus.mapper
Type: PVBusMapper.

cluster1.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster1.l2_cache
PV Cache.
Type: PVCache.

cluster1.l2_cache.upstream[0]
Type: PVBusSlave.

cluster1.l2_cache.upstream[10]
Type: PVBusSlave.

cluster1.l2_cache.upstream[11]
Type: PVBusSlave.

cluster1.l2_cache.upstream[12]
Type: PVBusSlave.

cluster1.l2_cache.upstream[13]
Type: PVBusSlave.

cluster1.l2_cache.upstream[14]
Type: PVBusSlave.

cluster1.l2_cache.upstream[15]
Type: PVBusSlave.

cluster1.l2_cache.upstream[16]
Type: PVBusSlave.

cluster1.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.44 FVP_Base_Cortex-A73x2

List of instances in FVP_Base_Cortex-A73x2.

FVP_Base_Cortex-A73x2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.virtio_rng**

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).**bp.virtio_rng.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice**

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).**bp.virtio_blockdevice.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice_labeller**Type: [Labeller](#).**bp.virtio_blockdevice_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.virtio_p9device**

virtio P9 server.

Type: [VirtioP9Device](#).**bp.virtio_p9device.mmio_slave**Type: [PVBusSlave](#).**bp.virtio_p9device.virtio_master**Type: [PVBusMaster](#).**bp.virtio_p9device_labeller**Type: [Labeller](#).**bp.virtio_p9device_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.vis**

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).**bp.vis.recorder**

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).**bp.vis.recorder.playbackDivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A73 Cluster CT model.

Type: [Cluster_ARM_Cortex-A73](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu1.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache
PV Cache.
Type: PVCache.

cluster0.cpu1.l1dcache.upstream[0]
Type: PVBusSlave.

cluster0.cpu1.l1icache
PV Cache.
Type: PVCache.

cluster0.cpu1.l1icache.upstream[0]
Type: PVBusSlave.

cluster0.ext_bus
Type: PVBusLogger.

cluster0.ext_bus.mapper
Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0
Type: [PVBusMaster](#).

pctl.utility_bus1
Type: [PVBusMaster](#).

pctl.utility_bus2
Type: [PVBusMaster](#).

pctl.utility_bus3
Type: [PVBusMaster](#).

12.45 FVP_Base_Cortex-A73x2-A53x4

List of instances in FVP_Base_Cortex-A73x2-A53x4.

FVP_Base_Cortex-A73x2-A53x4 instances

address_map_terminator
Type: [PVBusMapper](#).

bp
Peripherals and address map for the Base Platform.
Type: [BasePlatformPeripherals](#).

bp.Timer_0_1
ARM Dual-Timer Module(SP804).
Type: [SP804_Timer](#).

bp.Timer_0_1.busslave
Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.Timer_0_1.counter0
Internal component used by SP804 Timer module.
Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBusSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A73 Cluster CT model.

Type: [Cluster_ARM_Cortex-A73](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu1**

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).**cluster0.cpu1.UTLB**Type: [TLB](#).

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu1.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.ext_bus
Type: pVBusLogger.

cluster0.ext_bus.mapper
Type: pVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: pVCache.

cluster0.l2_cache.upstream[0]
Type: pVBusSlave.

cluster0.l2_cache.upstream[10]
Type: pVBusSlave.

cluster0.l2_cache.upstream[11]
Type: pVBusSlave.

cluster0.l2_cache.upstream[12]
Type: pVBusSlave.

cluster0.l2_cache.upstream[13]
Type: pVBusSlave.

cluster0.l2_cache.upstream[14]
Type: pVBusSlave.

cluster0.l2_cache.upstream[15]
Type: pVBusSlave.

cluster0.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

cluster1

ARM Cortex-A53 Cluster CT model.

Type: [cluster_ARM_Cortex-A53](#).

cluster1.AMU

Type: [PVBUSLogger](#).

cluster1.AMU.mapper

Type: [PVBUSMapper](#).

cluster1.DAP

Type: [PVBUSLogger](#).

cluster1.DAP.mapper

Type: [PVBUSMapper](#).

cluster1.DSU

Type: DSU.

cluster1.DSU.mpam_busslave

Type: PVBusSlave.

cluster1.MMAP

Type: PVBusLogger.

cluster1.MMAP.mapper

Type: PVBusMapper.

cluster1.acp_mapper

Type: PVBusMapper.

cluster1.cpu0

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: pvcache.

cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1.l1icache

PV Cache.

Type: pvcache.

cluster1.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu2.l1dcache

PV Cache.

Type: pvcache.

cluster1.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2.l1icache

PV Cache.

Type: pvcache.

cluster1.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu3

ARM Cortex-A53 CT model.

Type: ARM_Cortex-A53.

cluster1.cpu3.UTLB

Type: TLB.

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster1.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster1.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.ext_bus
Type: `PVBusLogger`.

cluster1.ext_bus.mapper
Type: `PVBusMapper`.

cluster1.gic_cpuiif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster1.l2_cache
PV Cache.
Type: `PVCache`.

cluster1.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_2

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_3

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.46 FVP_Base_Cortex-A73x4

List of instances in FVP_Base_Cortex-A73x4.

FVP_Base_Cortex-A73x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.coresight_modifier

Type: `PVBusMapper`.

bp.dmc

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc.bus_slave

Type: `PVBusSlave`.

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dmc_phy.bus_slave

Type: `PVBusSlave`.

bp.dram_limiter

Type: `PVBusMapper`.

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_local_dap_rom.bus_slave

Type: `PVBusSlave`.

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_ram.bus_slave

Type: `PVBusSlave`.

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.dummy_usb.bus_slave

Type: `PVBusSlave`.

bp.exclusive_monitor

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

bp.exclusive_monitor.bus_mapper

Type: `PVBusMapper`.

bp.fixed_security_map

Type: `PVBusMapper`.

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash0.map

Type: `PVBusMapper`.

bp.flash0.mbs

Type: `PVBusSlave`.

bp.flash0.rmbs

Type: `PVBusSlave`.

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.flash1.map

Type: `PVBusMapper`.

bp.flash1.mbs

Type: `PVBusSlave`.

bp.flash1.rmbs

Type: `PVBusSlave`.

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.generic_watchdog

ARM Generic Watchdog.

Type: `MemoryMappedGenericWatchdog`.

bp.generic_watchdog.busctlbase

Type: `PVBusSlave`.

bp.generic_watchdog.busrefbase

Type: `PVBusSlave`.

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCD`.

bp.hdlcd0.busmaster

Type: `PVBusMaster`.

bp.hdlcd0.busslave

Type: `PVBusSlave`.

bp.hdlcd0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbuslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly

or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slaveType: [PVBusSlave](#).**bp.pl011_uart0**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart0.busslave**Type: [PVBusSlave](#).**bp.pl011_uart0.clk_divider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.pl011_uart0.timer**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**bp.pl011_uart0.timer.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**bp.pl011_uart0.timer.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**bp.pl011_uart0.timer.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**bp.pl011_uart1**

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).**bp.pl011_uart1.busslave**Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).**bp.rl_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.rl_dram.bus_slave**Type: [PVBusSlave](#).**bp.rt_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.rt_dram.bus_slave**Type: [PVBusSlave](#).**bp.s_dram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.s_dram.bus_slave**Type: [PVBusSlave](#).**bp.secureDRAM**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.secureDRAM.bus_slave**Type: [PVBusSlave](#).**bp.secureSRAM**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.secureSRAM.bus_slave**Type: [PVBusSlave](#).**bp.secureSRAM_limiter**Type: [PVBusMapper](#).**bp.secureflash**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.secureflash.map**Type: [PVBusMapper](#).**bp.secureflash.mbs**Type: [PVBusSlave](#).**bp.secureflash.rmbs**Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBusSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: filter0.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: filter1.

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: filter2.

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: filter3.

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [VE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A73 Cluster CT model.

Type: `Cluster_ARM_Cortex-A73`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.acp_mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A73 CT model.

Type: `ARM_Cortex-A73`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu0.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3
ARM Cortex-A73 CT model.
Type: `ARM_Cortex-A73`.

cluster0.cpu3.UTLB
Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapperType: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.47 FVP_Base_Cortex-A73x4-A53x4

List of instances in FVP_Base_Cortex-A73x4-A53x4.

FVP_Base_Cortex-A73x4-A53x4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A73 Cluster CT model.

Type: [cluster_ARM_Cortex-A73](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A73 CT model.

Type: [ARM_Cortex-A73](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1

ARM Cortex-A73 CT model.

Type: `ARM_Cortex-A73`.

cluster0.cpu1.UTLB

Type: `TLB`.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu1.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A73 CT model.

Type: `ARM_Cortex-A73`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-A73 CT model.

Type: `ARM_Cortex-A73`.**cluster0.cpu3.UTLB**Type: `TLB`.**cluster0.cpu3.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu3.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu3.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu3.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.ext_bus**Type: `PVBusLogger`.**cluster0.ext_bus.mapper**Type: `PVBusMapper`.**cluster0.gic_cpuif_decoder_cluster**Type: `GICv3CPUInterfaceDecoder`.**cluster0.l2_cache**

PV Cache.

Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[10]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

cluster1

ARM Cortex-A53 Cluster CT model.

Type: [Cluster_ARM_Cortex-A53](#).

cluster1.AMU

Type: [PVBusLogger](#).

cluster1.AMU.mapper

Type: [PVBusMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: [DSU](#).

cluster1.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.acp_mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu0.UTLB

Type: [TLB](#).

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l1icache

PV Cache.

Type: `PVCache`.**cluster1.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu1**

ARM Cortex-A53 CT model.

Type: `ARM_Cortex-A53`.**cluster1.cpu1.UTLB**Type: `TLB`.**cluster1.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster1.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster1.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster1.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster1.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster1.cpu2**

ARM Cortex-A53 CT model.

Type: `ARM_Cortex-A53`.**cluster1.cpu2.UTLB**Type: `TLB`.**cluster1.cpu2.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster1.cpu2.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster1.cpu2.l1dcache**

PV Cache.

Type: `PVCache`.

cluster1.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu3

ARM Cortex-A53 CT model.

Type: [ARM_Cortex-A53](#).

cluster1.cpu3.UTLB

Type: [TLB](#).

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster1.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.ext_bus

Type: [PVBusLogger](#).

cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

cluster1.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[10]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[11]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[12]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[13]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[14]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[15]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_0_3**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_1**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_2**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_3**Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: `GICv3Distributor`.

pctl

Base Platforms Power Controller.

Type: `Base_PowerController`.

pctl.busslave

Type: `PVBusSlave`.

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.48 FVP_Base_Cortex-A75

List of instances in FVP_Base_Cortex-A75.

FVP_Base_Cortex-A75 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.ps2mouse.ps2_clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.psram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.psram.bus_slave

Type: `PVBusSlave`.

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

bp.refcounter.pvbus_control_s[0]

Type: `PVBusSlave`.

bp.refcounter.pvbus_read_s[0]

Type: `PVBusSlave`.

bp.reset_or

Or Gate.

Type: `OrGate`.

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rl_dram.bus_slave

Type: `PVBusSlave`.

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rt_dram.bus_slave

Type: `PVBusSlave`.

bp.s_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.s_dram.bus_slave

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A75 Cluster CT model.

Type: [Cluster_ARM_Cortex-A75](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A75 CT model.

Type: `ARM_Cortex-A75`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-A75 CT model.

Type: `ARM_Cortex-A75`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A75 CT model.

Type: `ARM_Cortex-A75`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A75 CT model.

Type: `ARM_Cortex-A75`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmasterType: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.49 FVP_Base_Cortex-A76

List of instances in FVP_Base_Cortex-A76.

FVP_Base_Cortex-A76 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A76 Cluster CT model.

Type: `Cluster_ARM_Cortex-A76`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.l3_flusher**Type: `AsyncCacheFlushUnit`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A76 CT model.

Type: [ARM_Cortex-A76](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A76 CT model.

Type: [ARM_Cortex-A76](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l2cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.cpu2
ARM Cortex-A76 CT model.
Type: [ARM_Cortex-A76](#).

cluster0.cpu2.UTLB
Type: [TLB](#).

cluster0.cpu2.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu2.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu2.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu2.l1icache
PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu3
ARM Cortex-A76 CT model.
Type: `ARM_Cortex-A76`.

cluster0.cpu3.UTLB
Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_2_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_3_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_tl**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).**pctl.utility_bus2**Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.50 FVP_Base_Cortex-A76AE

List of instances in FVP_Base_Cortex-A76AE.

FVP_Base_Cortex-A76AE instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtioblockdevice.dma_master

Type: `PVBusMaster`.

bp.virtioblockdevice_labeller

Type: `Labeller`.

bp.virtioblockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtiop9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtiop9device.mmio_slave

Type: `PVBusSlave`.

bp.virtiop9device.virtio_master

Type: `PVBusMaster`.

bp.virtiop9device_labeller

Type: `Labeller`.

bp.virtiop9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A76AE Cluster CT model.

Type: [Cluster_ARM_Cortex-A76AE](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A76AE CT model.

Type: `ARM_Cortex-A76AE`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-A76AE CT model.

Type: `ARM_Cortex-A76AE`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A76AE CT model.

Type: `ARM_Cortex-A76AE`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A76AE CT model.

Type: `ARM_Cortex-A76AE`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmasterType: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_2**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_2_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_3**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_3_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_t1**Type: [GICv3Distributor](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.51 FVP_Base_Cortex-A77

List of instances in FVP_Base_Cortex-A77.

FVP_Base_Cortex-A77 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A77 Cluster CT model.

Type: `Cluster_ARM_Cortex-A77`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.l3_flusher**Type: `AsyncCacheFlushUnit`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A77 CT model.

Type: [ARM_Cortex-A77](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A77 CT model.

Type: [ARM_Cortex-A77](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l2cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.cpu2
ARM Cortex-A77 CT model.
Type: [ARM_Cortex-A77](#).

cluster0.cpu2.UTLB
Type: [TLB](#).

cluster0.cpu2.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu2.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu2.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu2.l1icache
PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu3
ARM Cortex-A77 CT model.
Type: `ARM_Cortex-A77`.

cluster0.cpu3.UTLB
Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_2_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).**pctl.utility_bus2**Type: [PVBusMaster](#).

pctl.utility_bus3Type: [PVBusMaster](#).

12.52 FVP_Base_Cortex-A78

List of instances in FVP_Base_Cortex-A78.

FVP_Base_Cortex-A78 instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_2_3.busslave**Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A78 Cluster CT model.

Type: [Cluster_ARM_Cortex-A78](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A78 CT model.

Type: `ARM_Cortex-A78`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-A78 CT model.

Type: `ARM_Cortex-A78`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A78 CT model.

Type: `ARM_Cortex-A78`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A78 CT model.

Type: `ARM_Cortex-A78`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmasterType: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.53 FVP_Base_Cortex-A78AE

List of instances in FVP_Base_Cortex-A78AE.

FVP_Base_Cortex-A78AE instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

`bp.vis.recorder`

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

`bp.vis.recorder.playbackDivider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.vis.recorder.playbackTimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`bp.vis.recorder.playbackTimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.vis.recorder.playbackTimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.vis.recorder.playbackTimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.vis.recorder.recordingDivider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A78AE Cluster CT model.

Type: Cluster_ARM_Cortex-A78AE.

cluster0.AMU

Type: PVBusLogger.

cluster0.AMU.mapper

Type: PVBusMapper.

cluster0.DAP

Type: PVBusLogger.

cluster0.DAP.mapper

Type: PVBusMapper.

cluster0.DSU

Type: dsu.

cluster0.DSU.l3_flusher

Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: pvcache.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[1]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[2]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[3]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[4]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[5]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[6]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[7]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[8]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-A78AE CT model.

Type: [ARM_Cortex-A78AE](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-A78AE CT model.

Type: [ARM_Cortex-A78AE](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l2cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.cpu2
ARM Cortex-A78AE CT model.
Type: [ARM_Cortex-A78AE](#).

cluster0.cpu2.UTLB
Type: [TLB](#).

cluster0.cpu2.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu2.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu2.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu2.l1icache
PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu3
ARM Cortex-A78AE CT model.
Type: `ARM_Cortex-A78AE`.

cluster0.cpu3.UTLB
Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_2_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_3_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3Type: [PVBusMaster](#).

12.54 FVP_Base_Cortex-A78C

List of instances in FVP_Base_Cortex-A78C.

FVP_Base_Cortex-A78C instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_2_3.busslave**Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmb

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.psram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.psram.bus_slave

Type: `PVBusSlave`.

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

bp.refcounter.pvbus_control_s[0]

Type: `PVBusSlave`.

bp.refcounter.pvbus_read_s[0]

Type: `PVBusSlave`.

bp.reset_or

Or Gate.

Type: `OrGate`.

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rl_dram.bus_slave

Type: `PVBusSlave`.

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rt_dram.bus_slave

Type: `PVBusSlave`.

bp.s_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.s_dram.bus_slave

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-A78C Cluster CT model.

Type: [Cluster_ARM_Cortex-A78C](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-A78C CT model.

Type: `ARM_Cortex-A78C`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-A78C CT model.

Type: `ARM_Cortex-A78C`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-A78C CT model.

Type: `ARM_Cortex-A78C`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-A78C CT model.

Type: `ARM_Cortex-A78C`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmasterType: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.55 FVP_Base_Cortex-X1

List of instances in FVP_Base_Cortex-X1.

FVP_Base_Cortex-X1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-X1 Cluster CT model.

Type: `Cluster_ARM_Cortex-X1`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.l3_flusher**Type: `AsyncCacheFlushUnit`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-X1 CT model.

Type: [ARM_Cortex-X1](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-X1 CT model.

Type: [ARM_Cortex-X1](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1dcache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu1.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l1icache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu1.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu1.l2cache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu1.l2cache.upstream[1]
Type: [PVBusSlave](#).

cluster0.cpu2
ARM Cortex-X1 CT model.
Type: [ARM_Cortex-X1](#).

cluster0.cpu2.UTLB
Type: [TLB](#).

cluster0.cpu2.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu2.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu2.l1dcache.upstream[0]
Type: [PVBusSlave](#).

cluster0.cpu2.l1icache
PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu3
ARM Cortex-X1 CT model.
Type: `ARM_Cortex-X1`.

cluster0.cpu3.UTLB
Type: `TLB`.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_2_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).**pctl.utility_bus2**Type: [PVBusMaster](#).

pctl.utility_bus3Type: [PVBusMaster](#).

12.56 FVP_Base_Cortex-X1C

List of instances in FVP_Base_Cortex-X1C.

FVP_Base_Cortex-X1C instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_2_3.busslave**Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.psram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.psram.bus_slave

Type: `PVBusSlave`.

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

bp.refcounter.pvbus_control_s[0]

Type: `PVBusSlave`.

bp.refcounter.pvbus_read_s[0]

Type: `PVBusSlave`.

bp.reset_or

Or Gate.

Type: `OrGate`.

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rl_dram.bus_slave

Type: `PVBusSlave`.

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rt_dram.bus_slave

Type: `PVBusSlave`.

bp.s_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.s_dram.bus_slave

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-X1C Cluster CT model.

Type: [Cluster_ARM_Cortex-X1C](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.**cluster0.MMAP.mapper**Type: `PVBusMapper`.**cluster0.cpu0**

ARM Cortex-X1C CT model.

Type: `ARM_Cortex-X1C`.**cluster0.cpu0.UTLB**Type: `TLB`.**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu0.gicv3_cpu_if**Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-X1C CT model.

Type: `ARM_Cortex-X1C`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu1.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu1.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-X1C CT model.

Type: `ARM_Cortex-X1C`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu3

ARM Cortex-X1C CT model.

Type: `ARM_Cortex-X1C`.

cluster0.cpu3.UTLB

Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmasterType: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.57 FVP_Base_Cortex-X2

List of instances in FVP_Base_Cortex-X2.

FVP_Base_Cortex-X2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-X2 Cluster CT model.

Type: `Cluster_ARM_Cortex-X2`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.PPU_cluster**Type: `PPUv1`.**cluster0.DSU.PPU_cluster.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core0**Type: `PPUv1`.**cluster0.DSU.PPU_core0.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core1**Type: `PPUv1`.**cluster0.DSU.PPU_core1.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core10**Type: `PPUv1`.**cluster0.DSU.PPU_core10.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core11**Type: `PPUv1`.**cluster0.DSU.PPU_core11.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core2**Type: `PPUv1`.**cluster0.DSU.PPU_core2.busslave**Type: `PVBusSlave`.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core6

Type: PPUv1.

cluster0.DSU.PPU_core6.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core7

Type: PPUv1.

cluster0.DSU.PPU_core7.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core8

Type: PPUv1.

cluster0.DSU.PPU_core8.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core9

Type: PPUv1.

cluster0.DSU.PPU_core9.busslave

Type: PVBusSlave.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVPatch.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[10]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[11]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu1

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu1.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu10

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).

cluster0.cpu10.UTLB

Type: TLB.

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu10.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu10.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu10.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu10.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu10.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.cpu10.l2cache
PV Cache.
Type: pVCache.

cluster0.cpu10.l2cache.upstream[0]
Type: pVBusSlave.

cluster0.cpu10.l2cache.upstream[1]
Type: pVBusSlave.

cluster0.cpu11
ARM Cortex-X2 CT model.
Type: ARM_Cortex-X2.

cluster0.cpu11.UTLB
Type: TLB.

cluster0.cpu11.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster0.cpu11.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu11.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu11.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu11.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu11.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.cpu1.l2cache

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu2**

ARM Cortex-X2 CT model.

Type: `ARM_Cortex-X2`.**cluster0.cpu2.UTLB**Type: `TLB`.**cluster0.cpu2.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu2.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu2.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu2.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu3**

ARM Cortex-X2 CT model.

Type: `ARM_Cortex-X2`.**cluster0.cpu3.UTLB**Type: `TLB`.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu3.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu3.l1dcache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu3.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu3.l1icache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu3.l1icache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu3.l2cache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu3.l2cache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu3.l2cache.upstream[1]**Type: [PVBUSSlave](#).**cluster0.cpu4**

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).**cluster0.cpu4.UTLB**Type: [TLB](#).**cluster0.cpu4.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu4.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu4.l1dcache**

PV Cache.

Type: [pVCache](#).**cluster0.cpu4.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu4.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu4.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu4.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu4.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu5
ARM Cortex-X2 CT model.
Type: `ARM_Cortex-X2`.

cluster0.cpu5.UTLB
Type: `TLB`.

cluster0.cpu5.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu5.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu5.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu5.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu5.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu5.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu5.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu5.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu5.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu6
ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).

cluster0.cpu6.UTLB

Type: TLB.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu6.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu7

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).

cluster0.cpu7.UTLB

Type: TLB.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu7.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu7.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu7.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu7.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu7.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu8

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).

cluster0.cpu8.UTLB

Type: [TLB](#).

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu8.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu8.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache.upstream[1]Type: [PVBusSlave](#).**cluster0.cpu9**

ARM Cortex-X2 CT model.

Type: [ARM_Cortex-X2](#).**cluster0.cpu9.UTLB**

Type: TLB.

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu9.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu9.l1dcache

PV Cache.

Type: [PVCache](#).**cluster0.cpu9.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu9.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu9.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu9.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu9.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu9.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**

Type: GICv3CPUInterfaceDecoder.

cluster0_labellerType: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_10**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_10_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_11**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_11_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_2**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_2_0**Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_4

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_5

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.58 FVP_Base_Cortex-X3

List of instances in `FVP_Base_Cortex-X3`.

FVP_Base_Cortex-X3 instances

address_map_terminator

Type: `PVBusMapper`.

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smisc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: `filter0`.

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: `filter1`.

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: `filter2`.

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: `filter3`.

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-X3 Cluster CT model.

Type: [cluster_ARM_Cortex-X3](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUV1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core0

Type: [PPUV1](#).

cluster0.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core1

Type: [PPUV1](#).

cluster0.DSU.PPU_core1.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core10

Type: [PPUV1](#).

cluster0.DSU.PPU_core10.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core11

Type: [PPUV1](#).

cluster0.DSU.PPU_core11.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core6

Type: PPUv1.

cluster0.DSU.PPU_core6.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core7

Type: PPUv1.

cluster0.DSU.PPU_core7.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core8

Type: PPUv1.

cluster0.DSU.PPU_core8.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core9

Type: PPUv1.

cluster0.DSU.PPU_core9.busslave

Type: PVBusSlave.

cluster0.DSU.l3_flusher

Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: `PVCache`.

`cluster0.DSU.shared_cache.upstream[0]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[10]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[11]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[12]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[13]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[14]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[15]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[16]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[17]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[18]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[19]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[1]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[20]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[21]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[22]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[23]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[24]`

Type: `PVBusSlave`.

`cluster0.DSU.shared_cache.upstream[25]`

Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[2]Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[3]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[4]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[5]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[6]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[7]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[8]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[9]**Type: [PVBusSlave](#).**cluster0.DSU.utility_slave[0]**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-X3 CT model.

Type: [ARM_Cortex-X3](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-X3 CT model.

Type: `ARM_Cortex-X3`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache.upstream[1]**Type: `PVBusSlave`.

cluster0.cpu10

ARM Cortex-X3 CT model.

Type: [ARM_Cortex-X3](#).**cluster0.cpu10.UTLB**

Type: TLB.

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu10.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu10.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu10.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu10.l1icache**

PV Cache.

Type: PVCache.

cluster0.cpu10.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu10.l2cache**

PV Cache.

Type: PVCache.

cluster0.cpu10.l2cache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu10.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu11**

ARM Cortex-X3 CT model.

Type: [ARM_Cortex-X3](#).**cluster0.cpu11.UTLB**

Type: TLB.

cluster0.cpu11.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu11.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu11.l1dcache

PV Cache.

Type: `pVCache`.

cluster0.cpu11.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu11.l1icache

PV Cache.

Type: `pVCache`.

cluster0.cpu11.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu11.l2cache

PV Cache.

Type: `pVCache`.

cluster0.cpu11.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu11.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-X3 CT model.

Type: `ARM_Cortex-X3`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu2.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu2.l1dcache

PV Cache.

Type: `pVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l1icache

PV Cache.

Type: `pVCache`.

cluster0.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu2.l2cache

PV Cache.

Type: `pVCache`.

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-X3 CT model.

Type: [ARM_Cortex-X3](#).

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu3.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-X3 CT model.

Type: [ARM_Cortex-X3](#).

cluster0.cpu4.UTLB

Type: TLB.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu4.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu5

ARM Cortex-X3 CT model.

Type: [ARM_Cortex-X3](#).

cluster0.cpu5.UTLB

Type: [TLB](#).

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu5.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu5.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu5.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.l2cache

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu6**

ARM Cortex-X3 CT model.

Type: `ARM_Cortex-X3`.**cluster0.cpu6.UTLB**Type: `TLB`.**cluster0.cpu6.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu6.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu6.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu7**

ARM Cortex-X3 CT model.

Type: `ARM_Cortex-X3`.**cluster0.cpu7.UTLB**Type: `TLB`.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu7.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu7.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu8**

ARM Cortex-X3 CT model.

Type: `ARM_Cortex-X3`.**cluster0.cpu8.UTLB**Type: `TLB`.**cluster0.cpu8.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu8.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu8.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu8.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu8.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu8.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu8.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu8.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu8.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu9
ARM Cortex-X3 CT model.
Type: `ARM_Cortex-X3`.

cluster0.cpu9.UTLB
Type: `TLB`.

cluster0.cpu9.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu9.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu9.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_10_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_11

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_11_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_8
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_9
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0
Type: GICv3Redistributor.

gic_distributor.rd_tl
Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.59 FVP_Base_Cortex-X4

List of instances in FVP_Base_Cortex-X4.

FVP_Base_Cortex-X4 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.pl041_aaci.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.pl041_aaci.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: `PL050_KMI`.

`bp.pl050_kmi0.busslave`

Type: `PVBusSlave`.

`bp.pl050_kmi0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: `PL050_KMI`.

`bp.pl050_kmi1.busslave`

Type: `PVBusSlave`.

`bp.pl050_kmi1.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`bp.pl111_clcd`

ARM PrimeCell Color LCD Controller(PL111).

Type: `PL111_CLCD`.

`bp.pl111_clcd.pl11x_clcd`

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

`bp.pl111_clcd.pl11x_clcd.busmaster`

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtioblockdevice.dma_master

Type: `PVBusMaster`.

bp.virtioblockdevice_labeller

Type: `Labeller`.

bp.virtioblockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtiop9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtiop9device.mmio_slave

Type: `PVBusSlave`.

bp.virtiop9device.virtio_master

Type: `PVBusMaster`.

bp.virtiop9device_labeller

Type: `Labeller`.

bp.virtiop9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-X4 Cluster CT model.

Type: [Cluster_ARM_Cortex-X4](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUV1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBUSSlave](#).

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core10

Type: PPUv1.

cluster0.DSU.PPU_core10.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core11

Type: PPUv1.

cluster0.DSU.PPU_core11.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core12

Type: PPUv1.

cluster0.DSU.PPU_core12.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core13

Type: PPUv1.

cluster0.DSU.PPU_core13.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core6

Type: [PPUv1](#).

cluster0.DSU.PPU_core6.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core7

Type: [PPUv1](#).

cluster0.DSU.PPU_core7.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core8

Type: [PPUv1](#).

cluster0.DSU.PPU_core8.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core9

Type: [PPUv1](#).

cluster0.DSU.PPU_core9.busslave

Type: [PVBusSlave](#).

cluster0.DSU.l3_flusher

Type: [AsyncCacheFlushUnit](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[26]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[27]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[28]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[29]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[30]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[3]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBUSSlave](#).

cluster0.MMAP

Type: [PVBUSLogger](#).

cluster0.MMAP.mapper

Type: [PVBUSMapper](#).

cluster0.cpu0

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu0.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM Cortex-X4 CT model.

Type: `ARM_Cortex-X4`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu10**

ARM Cortex-X4 CT model.

Type: `ARM_Cortex-X4`.**cluster0.cpu10.UTLB**Type: `TLB`.

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu10.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu10.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu10.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu10.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu10.l1icache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu10.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu10.l2cache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu10.l2cache.upstream[1]**Type: [PVBUSSlave](#).**cluster0.cpu11**

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).**cluster0.cpu11.UTLB**Type: [TLB](#).**cluster0.cpu11.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu11.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu11.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu11.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu11.l1icache**

PV Cache.

Type: `PVCache`.

cluster0.cpu11.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu11.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu11.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu12
ARM Cortex-X4 CT model.
Type: `ARM_Cortex-X4`.

cluster0.cpu12.UTLB
Type: `TLB`.

cluster0.cpu12.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu12.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu12.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu12.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu12.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu12.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu12.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu12.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu12.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.cpu13
ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu13.UTLB

Type: TLB.

cluster0.cpu13.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu13.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu13.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu13.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu13.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu13.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu13.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu2

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu4.UTLB

Type: TLB.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu4.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu6**

ARM Cortex-X4 CT model.

Type: `ARM_Cortex-X4`.**cluster0.cpu6.UTLB**Type: `TLB`.**cluster0.cpu6.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu6.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu6.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu6.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu6.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu7

ARM Cortex-X4 CT model.

Type: `ARM_Cortex-X4`.

cluster0.cpu7.UTLB

Type: `TLB`.

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu7.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu7.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu7.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu7.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu8

ARM Cortex-X4 CT model.

Type: `ARM_Cortex-X4`.

cluster0.cpu8.UTLB

Type: `TLB`.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu8.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu8.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu8.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu8.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu9

ARM Cortex-X4 CT model.

Type: [ARM_Cortex-X4](#).

cluster0.cpu9.UTLB

Type: [TLB](#).

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu9.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu9.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu9.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu9.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu9.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu9.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu9.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**Type: [GICv3CPUInterfaceDecoder](#).**cluster0_labeller**Type: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_11
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_11_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_12
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_12_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_13
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_13_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.60 FVP_Base_Cortex-X925

List of instances in FVP_Base_Cortex-X925.

FVP_Base_Cortex-X925 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBusSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBUSSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBUSSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBUSSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-X925 Cluster CT model.

Type: [Cluster_ARM_Cortex-X925](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core10

Type: PPUv1.

cluster0.DSU.PPU_core10.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core11

Type: PPUv1.

cluster0.DSU.PPU_core11.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core12

Type: PPUv1.

cluster0.DSU.PPU_core12.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core13

Type: PPUv1.

cluster0.DSU.PPU_core13.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4
Type: PPUv1.

cluster0.DSU.PPU_core4.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core5
Type: PPUv1.

cluster0.DSU.PPU_core5.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core6
Type: PPUv1.

cluster0.DSU.PPU_core6.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core7
Type: PPUv1.

cluster0.DSU.PPU_core7.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core8
Type: PPUv1.

cluster0.DSU.PPU_core8.busslave
Type: PVBusSlave.

cluster0.DSU.PPU_core9
Type: PPUv1.

cluster0.DSU.PPU_core9.busslave
Type: PVBusSlave.

cluster0.DSU.l3_flusher
Type: AsyncCacheFlushUnit.

cluster0.DSU.mpam_busslave
Type: PVBusSlave.

cluster0.DSU.shared_cache
PV Cache.
Type: PVCache.

cluster0.DSU.shared_cache.upstream[0]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[10]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[11]
Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[12]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[13]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[14]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[15]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[16]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[17]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[18]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[19]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[20]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[21]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[22]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[23]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[24]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[25]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[26]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[27]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[28]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[29]
Type: [PVBUSSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[30]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu1

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu1.UTLB

Type: [TLB](#).

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu10

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu10.UTLB

Type: TLB.

cluster0.cpu10.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu10.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu10.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu10.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu10.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu11

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu11.UTLB

Type: TLB.

cluster0.cpu11.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu11.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu11.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu11.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu11.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu11.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu11.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu11.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu11.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu12**

ARM Cortex-X925 CT model.

Type: `ARM_Cortex-X925`.**cluster0.cpu12.UTLB**Type: `TLB`.**cluster0.cpu12.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster0.cpu12.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu12.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu12.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu12.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu12.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu12.l2cache**

PV Cache.

Type: `PVCache`.

cluster0.cpu12.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu12.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu13

ARM Cortex-X925 CT model.

Type: `ARM_Cortex-X925`.

cluster0.cpu13.UTLB

Type: `TLB`.

cluster0.cpu13.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu13.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu13.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu13.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu13.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu13.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu13.l2cache

PV Cache.

Type: `PVCache`.

cluster0.cpu13.l2cache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu13.l2cache.upstream[1]

Type: `PVBusSlave`.

cluster0.cpu2

ARM Cortex-X925 CT model.

Type: `ARM_Cortex-X925`.

cluster0.cpu2.UTLB

Type: `TLB`.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu4

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu4.UTLB

Type: [TLB](#).

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu4.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu4.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu4.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu4.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu5

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).

cluster0.cpu5.UTLB

Type: TLB.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu5.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu5.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu5.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu5.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu5.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu5.l2cache.upstream[0]

Type: PVBusSlave.

cluster0.cpu5.l2cache.upstream[1]

Type: PVBusSlave.

cluster0.cpu6

ARM Cortex-X925 CT model.

Type: ARM_Cortex-X925.

cluster0.cpu6.UTLB

Type: TLB.

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu6.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu6.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu6.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu6.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l2cache.upstream[1]**Type: `PVBusSlave`.**cluster0.cpu7**

ARM Cortex-X925 CT model.

Type: `ARM_Cortex-X925`.**cluster0.cpu7.UTLB**Type: `TLB`.**cluster0.cpu7.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu7.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu7.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache**

PV Cache.

Type: `PVCache`.**cluster0.cpu7.l2cache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7.l2cache.upstream[1]**Type: `PVBusSlave`.

cluster0.cpu8

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).**cluster0.cpu8.UTLB**

Type: TLB.

cluster0.cpu8.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu8.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu8.l1dcache

PV Cache.

Type: pVCache.

cluster0.cpu8.l1dcache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu8.l1icache**

PV Cache.

Type: pVCache.

cluster0.cpu8.l1icache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu8.l2cache**

PV Cache.

Type: pVCache.

cluster0.cpu8.l2cache.upstream[0]Type: [PVBUSSlave](#).**cluster0.cpu8.l2cache.upstream[1]**Type: [PVBUSSlave](#).**cluster0.cpu9**

ARM Cortex-X925 CT model.

Type: [ARM_Cortex-X925](#).**cluster0.cpu9.UTLB**

Type: TLB.

cluster0.cpu9.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu9.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu9.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu9.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu9.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu9.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmaster
Type: `PVBusMaster`.

gic_distributor
GIC Interrupt Redistribution Infrastructure component.
Type: `GIC_IRI`.

gic_distributor.rd_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_10_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_11
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_11_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_12
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_12_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_13
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_13_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_5

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_6

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_8

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_8_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_9

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_9_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.61 FVP_Base_Neoverse-E1

List of instances in FVP_Base_Neoverse-E1.

FVP_Base_Neoverse-E1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-E1 Cluster CT model.

Type: `cluster_ARM_Neoverse-E1`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.l3_flusher**Type: `AsyncCacheFlushUnit`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[10]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[11]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[12]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[13]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[14]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[15]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[16]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[17]**Type: `PVBusSlave`.

cluster0.DSU.shared_cache.upstream[1]Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[2]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[3]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[4]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[5]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[6]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[7]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[8]**Type: [PVBUSSlave](#).**cluster0.DSU.shared_cache.upstream[9]**Type: [PVBUSSlave](#).**cluster0.MMAP**Type: [PVBUSLogger](#).**cluster0.MMAP.mapper**Type: [PVBUSMapper](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.thread0**

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).**cluster0.cpu0.thread0.UTLB**Type: [TLB](#).**cluster0.cpu0.thread0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.thread0.l1dcache**Type: [PVCACHE](#).**cluster0.cpu0.thread0.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster0.cpu0.thread0.l1icache**Type: [PVCACHE](#).

cluster0.cpu0.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l2cache

Type: [PVCACHE](#).

cluster0.cpu0.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu0.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu0.thread1.UTLB

Type: [TLB](#).

cluster0.cpu0.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TLBCADI](#).

cluster0.cpu1.thread0

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu1.thread0.UTLB

Type: [TLB](#).

cluster0.cpu1.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.thread0.l1dcache

Type: [PVCACHE](#).

cluster0.cpu1.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread0.l1icache

Type: [PVCACHE](#).

cluster0.cpu1.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread0.l2cache

Type: [PVCACHE](#).

cluster0.cpu1.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu1.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu1.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu1.thread1.UTLB

Type: TLB.

cluster0.cpu1.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.thread0

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu2.thread0.UTLB

Type: TLB.

cluster0.cpu2.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.thread0.l1dcache

Type: PVPCache.

cluster0.cpu2.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.thread0.l1icache

Type: PVPCache.

cluster0.cpu2.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.thread0.l2cache

Type: PVPCache.

cluster0.cpu2.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu2.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu2.thread1.UTLB

Type: TLB.

cluster0.cpu2.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu3.thread0

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu3.thread0.UTLB

Type: TLB.

cluster0.cpu3.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.thread0.l1dcache

Type: PVCache.

cluster0.cpu3.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l1icache

Type: PVCache.

cluster0.cpu3.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l2cache

Type: PVCache.

cluster0.cpu3.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu3.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu3.thread1.UTLB

Type: TLB.

cluster0.cpu3.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu4.thread0

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu4.thread0.UTLB

Type: TLB.

cluster0.cpu4.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu4.thread0.l1dcache

Type: PVPCache.

cluster0.cpu4.thread0.l1dcache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu4.thread0.l1icache

Type: PVPCache.

cluster0.cpu4.thread0.l1icache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu4.thread0.l2cache

Type: PVPCache.

cluster0.cpu4.thread0.l2cache.upstream[0]

Type: PVPBusSlave.

cluster0.cpu4.thread0.l2cache.upstream[1]

Type: PVPBusSlave.

cluster0.cpu4.thread1

ARM Neoverse-E1 CT model.

Type: ARM_Neoverse-E1.

cluster0.cpu4.thread1.UTLB

Type: TLB.

cluster0.cpu4.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu5.thread0

ARM Neoverse-E1 CT model.

Type: ARM_Neoverse-E1.

cluster0.cpu5.thread0.UTLB

Type: TLB.

cluster0.cpu5.thread0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu5.thread0.l1dcache

Type: PVPCache.

cluster0.cpu5.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu5.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu5.thread0.l2cache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread0.l2cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.cpu5.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu5.thread1.UTLB

Type: [TLB](#).

cluster0.cpu5.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu6.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu6.thread0

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu6.thread0.UTLB

Type: [TLB](#).

cluster0.cpu6.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu6.thread0.l1dcache

Type: [PVCache](#).

cluster0.cpu6.thread0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu6.thread0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu6.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu6.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu6.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu6.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu6.thread1.UTLB

Type: [TLB](#).

cluster0.cpu6.thread1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu7.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu7.thread0

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).

cluster0.cpu7.thread0.UTLB

Type: [TLB](#).

cluster0.cpu7.thread0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu7.thread0.l1dcache

Type: [PVCache](#).

cluster0.cpu7.thread0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.thread0.l1icache

Type: [PVCache](#).

cluster0.cpu7.thread0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.thread0.l2cache

Type: [PVCache](#).

cluster0.cpu7.thread0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu7.thread0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu7.thread1

ARM Neoverse-E1 CT model.

Type: [ARM_Neoverse-E1](#).**cluster0.cpu7.thread1.UTLB**

Type: TLB.

cluster0.cpu7.thread1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.ext_busType: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**

Type: GICv3CPUInterfaceDecoder.

cluster0_labellerType: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_0_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6_1
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_7_1

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).**pctl.timer_reset.timer.thread**

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).**pctl.timer_reset.timer.thread_event**

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**pctl.utility_bus0**Type: [PVBusMaster](#).**pctl.utility_bus1**Type: [PVBusMaster](#).

pctl.utility_bus2Type: [PVBusMaster](#).**pctl.utility_bus3**Type: [PVBusMaster](#).

12.62 FVP_Base_Neoverse-N1

List of instances in FVP_Base_Neoverse-N1.

FVP_Base_Neoverse-N1 instances

address_map_terminatorType: [PVBusMapper](#).**bp**

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).**bp.Timer_0_1**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).**bp.Timer_0_1.busslave**Type: [PVBusSlave](#).**bp.Timer_0_1.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.Timer_0_1.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_0_1.counter1**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**bp.Timer_2_3**

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBUSSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBUSSlave](#).

bp.dram_limiterType: [PVBusMapper](#).**bp.dummy_local_dap_rom**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_local_dap_rom.bus_slave**Type: [PVBusSlave](#).**bp.dummy_ram**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_ram.bus_slave**Type: [PVBusSlave](#).**bp.dummy_usb**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**bp.dummy_usb.bus_slave**Type: [PVBusSlave](#).**bp.exclusive_monitor**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).**bp.exclusive_monitor.bus_mapper**Type: [PVBusMapper](#).**bp.fixed_security_map**Type: [PVBusMapper](#).**bp.flash0**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash0.map**Type: [PVBusMapper](#).**bp.flash0.mbs**Type: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hd1cd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbuslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1
Type: filter1.

bp.tzc_400.filter1.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [VE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [VEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.

Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

cci400.cciregisters.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

cci400.cciregisters.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

cci400.cciregisters.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

clockdivider0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

cluster0

ARM Neoverse-N1 Cluster CT model.

Type: `Cluster_ARM_Neoverse-N1`.

cluster0.AMU

Type: `PVBusLogger`.

cluster0.AMU.mapper

Type: `PVBusMapper`.

cluster0.DAP

Type: `PVBusLogger`.

cluster0.DAP.mapper

Type: `PVBusMapper`.

cluster0.DSU

Type: `DSU`.

cluster0.DSU.13_flusher

Type: `AsyncCacheFlushUnit`.

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Neoverse-N1 CT model.

Type: [ARM_Neoverse-N1](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu0.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu0.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu0.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.cpu0.l2cache
PV Cache.
Type: pVCache.

cluster0.cpu0.l2cache.upstream[0]
Type: pVBusSlave.

cluster0.cpu0.l2cache.upstream[1]
Type: pVBusSlave.

cluster0.cpu1
ARM Neoverse-N1 CT model.
Type: ARM_Neoverse-N1.

cluster0.cpu1.UTLB
Type: TLB.

cluster0.cpu1.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1dcache.upstream[0]
Type: pVBusSlave.

cluster0.cpu1.l1icache
PV Cache.
Type: pVCache.

cluster0.cpu1.l1icache.upstream[0]
Type: pVBusSlave.

cluster0.cpu1.l2cache

PV Cache.

Type: [PVCache](#).**cluster0.cpu1.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu1.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu2**

ARM Neoverse-N1 CT model.

Type: [ARM_Neoverse-N1](#).**cluster0.cpu2.UTLB**

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).**cluster0.cpu2.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu2.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu2.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu2.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu2.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu2.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu2.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu2.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu3**

ARM Neoverse-N1 CT model.

Type: [ARM_Neoverse-N1](#).**cluster0.cpu3.UTLB**

Type: TLB.

cluster0.cpu3.dtlb
TLB - instruction, data or unified.
Type: `TlbCadi`.

cluster0.cpu3.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu3.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_2

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_2_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_0_3

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_3_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.63 FVP_Base_Neoverse-N2

List of instances in FVP_Base_Neoverse-N2.

FVP_Base_Neoverse-N2 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PvBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PvBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PvBusMapper](#).

bp.utility_bus_map0

Type: [PvBusMapper](#).

bp.utility_bus_map1

Type: [PvBusMapper](#).

bp.utility_bus_map2

Type: [PvBusMapper](#).

bp.utility_bus_map3

Type: [PvBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PvBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-N2 Cluster CT model.

Type: `cluster_ARM_Neoverse-N2`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.PPU_cluster**Type: `PPUv1`.**cluster0.DSU.PPU_cluster.busslave**Type: `PVBusSlave`.**cluster0.DSU.PPU_core0**Type: `PPUv1`.**cluster0.DSU.PPU_core0.busslave**Type: `PVBusSlave`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.utility_slave[0]**Type: `PVBusSlave`.**cluster0.MMAP**Type: `PVBusLogger`.

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Neoverse-N2 CT model.

Type: [ARM_Neoverse-N2](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: PVCache.

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.64 FVP_Base_Neoverse-N2x1-Neoverse-N2x1

List of instances in `FVP_Base_Neoverse-N2x1-Neoverse-N2x1`.

FVP_Base_Neoverse-N2x1-Neoverse-N2x1 instances

address_map_terminator

Type: `PVBusMapper`.

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_0_1.busslave

Type: `PVBusSlave`.

bp.Timer_0_1.clk_div0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_mapType: [PVBusMapper](#).**bp.flash0**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash0.map**Type: [PVBusMapper](#).**bp.flash0.mbs**Type: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBUSSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-N2 Cluster CT model.

Type: [cluster_ARM_Neoverse-N2](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUV1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core0

Type: [PPUV1](#).

cluster0.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Neoverse-N2 CT model.

Type: [ARM_Neoverse-N2](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0_labeller
Type: Labeller.

cluster0_labeller.pvbusmodifier
Type: PVBusMapper.

cluster1
ARM Neoverse-N2 Cluster CT model.
Type: cluster_ARM_Neoverse-N2.

cluster1.AMU
Type: PVBusLogger.

cluster1.AMU.mapper
Type: PVBusMapper.

cluster1.DAP
Type: PVBusLogger.

cluster1.DAP.mapper
Type: PVBusMapper.

cluster1.DSU
Type: DSU.

cluster1.DSU.PPU_cluster
Type: PPUv1.

cluster1.DSU.PPU_cluster.busslave
Type: PVBusSlave.

cluster1.DSU.PPU_core0
Type: PPUv1.

cluster1.DSU.PPU_core0.busslave
Type: PVBusSlave.

cluster1.DSU.mpam_busslave
Type: PVBusSlave.

cluster1.DSU.shared_cache
PV Cache.
Type: Pvcache.

cluster1.DSU.shared_cache.upstream[0]
Type: PVBusSlave.

cluster1.DSU.shared_cache.upstream[1]
Type: PVBusSlave.

cluster1.DSU.shared_cache.upstream[2]
Type: PVBusSlave.

cluster1.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Neoverse-N2 CT model.

Type: [ARM_Neoverse-N2](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.ext_bus

Type: [PVBusLogger](#).

cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

cluster1.gic_cpuif_decoder_cluster
Type: [GICv3CPUInterfaceDecoder](#).

cluster1_labeller
Type: [Labeller](#).

cluster1_labeller.pvbusmodifier
Type: [PVBusMapper](#).

dapmemlogger
Bus Logger.
Type: [PVBusLogger](#).

dapmemlogger.mapper
Type: [PVBusMapper](#).

elfloader
ELF loader component.
Type: [ElfLoader](#).

elfloader.pvbus_busmaster
Type: [PVBusMaster](#).

gic_distributor
GIC Interrupt Redistribution Infrastructure component.
Type: [GIC_IRI](#).

gic_distributor.rd_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0
Type: [GICv3Redistributor](#).

gic_distributor.rd_0_1
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0
Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0_0
Type: [GICv3Redistributor](#).

gic_distributor.rd_tl
Type: [GICv3Distributor](#).

pctl
Base Platforms Power Controller.
Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.65 FVP_Base_Neoverse-N3

List of instances in FVP_Base_Neoverse-N3.

FVP_Base_Neoverse-N3 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labellerType: [Labeller](#).**bp.hdlcd0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.hostbridge**

Host Socket Interface Component.

Type: [HostBridge](#).**bp.lcd_security_map**Type: [PVBusMapper](#).**bp.ls64_testing_fifo**

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).**bp.ls64_testing_fifo.pvbusslave**Type: [PVBusSlave](#).**bp.mmc**

Generic Multimedia Card.

Type: [MMC](#).**bp.mmc.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**bp.mmc.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**bp.mmc.timer.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**bp.mmc.timer.timer.thread_event**

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-N3 Cluster CT model.

Type: [Cluster_ARM_Neoverse-N3](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUv1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core0Type: [PPUv1](#).**cluster0.DSU.PPU_core0.busslave**Type: [PVBusSlave](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache**

PV Cache.

Type: [PVCache](#).**cluster0.DSU.shared_cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[2]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[3]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[4]**Type: [PVBusSlave](#).**cluster0.DSU.utility_slave[0]**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Neoverse-N3 CT model.

Type: [ARM_Neoverse-N3](#).**cluster0.cpu0.UTLB**Type: [TLB](#).**cluster0.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.66 FVP_Base_Neoverse-N3x1-Neoverse-N3x1

List of instances in FVP_Base_Neoverse-N3x1-Neoverse-N3x1.

FVP_Base_Neoverse-N3x1-Neoverse-N3x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBUSSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBUSSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smisc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1
Type: filter1.

bp.tzc_400.filter1.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.

Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

cci400.cciregisters.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

cci400.cciregisters.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

cci400.cciregisters.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

clockdivider0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

clockdivider1

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

cluster0

ARM Neoverse-N3 Cluster CT model.

Type: `Cluster_ARM_Neoverse-N3`.

cluster0.AMU

Type: `PVBusLogger`.

cluster0.AMU.mapper

Type: `PVBusMapper`.

cluster0.DAP

Type: `PVBusLogger`.

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUv1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core0

Type: [PPUv1](#).

cluster0.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Neoverse-N3 CT model.

Type: [ARM_Neoverse-N3](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if
Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache
PV Cache.
Type: [pvcache](#).

cluster0.cpu0.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu0.l1icache
PV Cache.
Type: [pvcache](#).

cluster0.cpu0.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache
PV Cache.
Type: [pvcache](#).

cluster0.cpu0.l2cache.upstream[0]
Type: [PVBUSSlave](#).

cluster0.cpu0.l2cache.upstream[1]
Type: [PVBUSSlave](#).

cluster0.ext_bus
Type: [PVBUSLogger](#).

cluster0.ext_bus.mapper
Type: [PVBUSMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: [GICv3CPUInterfaceDecoder](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBUSMapper](#).

cluster1
ARM Neoverse-N3 Cluster CT model.
Type: [cluster_ARM_Neoverse-N3](#).

cluster1.AMU
Type: [PVBUSLogger](#).

cluster1.AMU.mapper
Type: [PVBUSMapper](#).

cluster1.DAP

Type: [PVBusLogger](#).

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: [DSU](#).

cluster1.DSU.PPU_cluster

Type: [PPUv1](#).

cluster1.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster1.DSU.PPU_core0

Type: [PPUv1](#).

cluster1.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster1.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster1.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Neoverse-N3 CT model.

Type: [ARM_Neoverse-N3](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l2cache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l2cache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l2cache.upstream[1]

Type: PVBusSlave.

cluster1.ext_bus

Type: PVBusLogger.

cluster1.ext_bus.mapper

Type: PVBusMapper.

cluster1.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster1_labeller

Type: Labeller.

cluster1_labeller.pvbusmodifier

Type: PVBusMapper.

dapmemlogger

Bus Logger.

Type: PVBusLogger.

dapmemlogger.mapper

Type: PVBusMapper.

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.67 FVP_Base_Neoverse-V1

List of instances in FVP_Base_Neoverse-V1.

FVP_Base_Neoverse-V1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBusSlave](#).

bp.ap_refclk.busbase1
Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBusSlave](#).

bp.ap_refclk.busbase11
Type: [PVBusSlave](#).

bp.ap_refclk.busbase12
Type: [PVBusSlave](#).

bp.ap_refclk.busbase13
Type: [PVBusSlave](#).

bp.ap_refclk.busbase14
Type: [PVBusSlave](#).

bp.ap_refclk.busbase15
Type: [PVBusSlave](#).

bp.ap_refclk.busbase2
Type: [PVBusSlave](#).

bp.ap_refclk.busbase3
Type: [PVBusSlave](#).

bp.ap_refclk.busbase4
Type: [PVBusSlave](#).

bp.ap_refclk.busbase5
Type: [PVBusSlave](#).

bp.ap_refclk.busbase6
Type: [PVBusSlave](#).

bp.ap_refclk.busbase7
Type: [PVBusSlave](#).

bp.ap_refclk.busbase8
Type: [PVBusSlave](#).

bp.ap_refclk.busbase9
Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-V1 Cluster CT model.

Type: `cluster_ARM_Neoverse-V1`.**cluster0.AMU**Type: `PVBusLogger`.**cluster0.AMU.mapper**Type: `PVBusMapper`.**cluster0.DAP**Type: `PVBusLogger`.**cluster0.DAP.mapper**Type: `PVBusMapper`.**cluster0.DSU**Type: `DSU`.**cluster0.DSU.mpam_busslave**Type: `PVBusSlave`.**cluster0.DSU.shared_cache**

PV Cache.

Type: `PVCache`.**cluster0.DSU.shared_cache.upstream[0]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[1]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[2]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[3]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[4]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[5]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[6]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[7]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[8]**Type: `PVBusSlave`.**cluster0.DSU.shared_cache.upstream[9]**Type: `PVBusSlave`.

cluster0.MMAPType: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Neoverse-V1 CT model.

Type: [ARM_Neoverse-V1](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu1**

ARM Neoverse-V1 CT model.

Type: [ARM_Neoverse-V1](#).**cluster0.cpu1.UTLB**

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Neoverse-V1 CT model.

Type: [ARM_Neoverse-V1](#).

cluster0.cpu2.UTLB

Type: [TLB](#).

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l2cache

PV Cache.

Type: [PVCache](#).**cluster0.cpu2.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu2.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.cpu3**

ARM Neoverse-V1 CT model.

Type: [ARM_Neoverse-V1](#).**cluster0.cpu3.UTLB**

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).**cluster0.cpu3.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu3.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu3.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu3.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu3.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu3.l2cache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu3.l2cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu3.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0_labeller
Type: Labeller.

cluster0_labeller.pvbusmodifier
Type: PVBusMapper.

dapmemlogger
Bus Logger.
Type: PVBusLogger.

dapmemlogger.mapper
Type: PVBusMapper.

elfloader
ELF loader component.
Type: ElfLoader.

elfloader.pvbus_busmaster
Type: PVBusMaster.

gic_distributor
GIC Interrupt Redistribution Infrastructure component.
Type: GIC_IRI.

gic_distributor.rd_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.68 FVP_Base_Neoverse-V2x1-Neoverse-V2x1

List of instances in FVP_Base_Neoverse-V2x1-Neoverse-V2x1.

FVP_Base_Neoverse-V2x1-Neoverse-V2x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hd1cd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hd1cd0_labeller

Type: [Labeller](#).

bp.hd1cd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

bp.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

bp.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

bp.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl111_clcd_labeller

Type: `Labeller`.

bp.pl111_clcd_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

bp.pl180_mci.busslave

Type: `PVBusSlave`.

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.psram.bus_slave`

Type: `PVBusSlave`.

`bp.refcounter`

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

`bp.refcounter.pvbus_control_s[0]`

Type: `PVBusSlave`.

`bp.refcounter.pvbus_read_s[0]`

Type: `PVBusSlave`.

`bp.reset_or`

Or Gate.

Type: `OrGate`.

`bp.rl_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rl_dram.bus_slave`

Type: `PVBusSlave`.

`bp.rt_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.rt_dram.bus_slave`

Type: `PVBusSlave`.

`bp.s_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.s_dram.bus_slave`

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtioblockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtioblockdevice_labeller

Type: [Labeller](#).

bp.virtioblockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtiop9device.virtio_master

Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-V2 Cluster CT model.

Type: [Cluster_ARM_Neoverse-V2](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVCache.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[1]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[2]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[3]

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Neoverse-V2 CT model.

Type: ARM_Neoverse-V2.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache
PV Cache.
Type: PVCache.

cluster0.cpu0.l1dcache.upstream[0]
Type: PVBusSlave.

cluster0.cpu0.l1icache
PV Cache.
Type: PVCache.

cluster0.cpu0.l1icache.upstream[0]
Type: PVBusSlave.

cluster0.cpu0.l2cache
PV Cache.
Type: PVCache.

cluster0.cpu0.l2cache.upstream[0]
Type: PVBusSlave.

cluster0.cpu0.l2cache.upstream[1]
Type: PVBusSlave.

cluster0.ext_bus
Type: PVBusLogger.

cluster0.ext_bus.mapper
Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0_labeller
Type: Labeller.

cluster0_labeller.pvbusmodifier
Type: PVBusMapper.

cluster1
ARM Neoverse-V2 Cluster CT model.
Type: Cluster_ARM_Neoverse-V2.

cluster1.AMU
Type: PVBusLogger.

cluster1.AMU.mapper
Type: PVBusMapper.

cluster1.DAP
Type: PVBusLogger.

cluster1.DAP.mapper

Type: [PVBusMapper](#).

cluster1.DSU

Type: [DSU](#).

cluster1.DSU.PPU_cluster

Type: [PPUv1](#).

cluster1.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster1.DSU.PPU_core0

Type: [PPUv1](#).

cluster1.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster1.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster1.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Neoverse-V2 CT model.

Type: [ARM_Neoverse-V2](#).

cluster1.cpu0.UTLB

Type: [TLB](#).

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster1.cpu0.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster1.cpu0.l1dcache
PV Cache.
Type: `PVCache`.

cluster1.cpu0.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu0.l1icache
PV Cache.
Type: `PVCache`.

cluster1.cpu0.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu0.l2cache
PV Cache.
Type: `PVCache`.

cluster1.cpu0.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster1.cpu0.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster1.ext_bus
Type: `PVBusLogger`.

cluster1.ext_bus.mapper
Type: `PVBusMapper`.

cluster1.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster1_labeller
Type: `Labeller`.

cluster1_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_t1

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.69 FVP_Base_Neoverse-V3

List of instances in FVP_Base_Neoverse-V3.

FVP_Base_Neoverse-V3 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout
SDL based Audio Output for PL041_AACI.
Type: [AudioOut_SDL](#).

bp.clock100Hz
A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.
Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smisc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-V3 Cluster CT model.

Type: [cluster_ARM_Neoverse-V3](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVPatch.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[1]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[2]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[3]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[4]

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Neoverse-V3 CT model.

Type: ARM_Neoverse-V3.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu0.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.70 FVP_Base_Neoverse-V3AE

List of instances in FVP_Base_Neoverse-V3AE.

FVP_Base_Neoverse-V3AE instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbsType: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-V3AE Cluster CT model.

Type: [Cluster_ARM_Neoverse-V3AE](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.mpam_busslave

Type: PVBusSlave.

cluster0.DSU.shared_cache

PV Cache.

Type: PVPatch.

cluster0.DSU.shared_cache.upstream[0]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[1]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[2]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[3]

Type: PVBusSlave.

cluster0.DSU.shared_cache.upstream[4]

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Neoverse-V3AE CT model.

Type: ARM_Neoverse-V3AE.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu0.gicv3_cpu_if
Type: `GICv3CPUInterface`.

cluster0.cpu0.l1dcache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l2cache
PV Cache.
Type: `PVCache`.

cluster0.cpu0.l2cache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu0.l2cache.upstream[1]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0_labeller
Type: `Labeller`.

cluster0_labeller.pvbusmodifier
Type: `PVBusMapper`.

dapmemlogger
Bus Logger.
Type: `PVBusLogger`.

dapmemlogger.mapper
Type: `PVBusMapper`.

elfloader
ELF loader component.
Type: `ElfLoader`.

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_t1

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.71 FVP_Base_Neoverse-V3AEx1-Neoverse-V3AEx1

List of instances in FVP_Base_Neoverse-V3AEx1-Neoverse-V3AEx1.

FVP_Base_Neoverse-V3AEx1-Neoverse-V3AEx1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12Type: [PVBusSlave](#).**bp.ap_refclk.busbase13**Type: [PVBusSlave](#).**bp.ap_refclk.busbase14**Type: [PVBusSlave](#).**bp.ap_refclk.busbase15**Type: [PVBusSlave](#).**bp.ap_refclk.busbase2**Type: [PVBusSlave](#).**bp.ap_refclk.busbase3**Type: [PVBusSlave](#).**bp.ap_refclk.busbase4**Type: [PVBusSlave](#).**bp.ap_refclk.busbase5**Type: [PVBusSlave](#).**bp.ap_refclk.busbase6**Type: [PVBusSlave](#).**bp.ap_refclk.busbase7**Type: [PVBusSlave](#).**bp.ap_refclk.busbase8**Type: [PVBusSlave](#).**bp.ap_refclk.busbase9**Type: [PVBusSlave](#).**bp.ap_refclk.busctlbase**Type: [PVBusSlave](#).**bp.audioout**

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).**bp.clock100Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.clock24MHz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbsType: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.virtio_rng**

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).**bp.virtio_rng.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice**

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).**bp.virtio_blockdevice.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice_labeller**Type: [Labeller](#).**bp.virtio_blockdevice_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.virtio_p9device**

virtio P9 server.

Type: [VirtioP9Device](#).**bp.virtio_p9device.mmio_slave**Type: [PVBusSlave](#).**bp.virtio_p9device.virtio_master**Type: [PVBusMaster](#).**bp.virtio_p9device_labeller**Type: [Labeller](#).**bp.virtio_p9device_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.vis**

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).**bp.vis.recorder**

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).**bp.vis.recorder.playbackDivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-V3AE Cluster CT model.

Type: [Cluster_ARM_Neoverse-V3AE](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAPType: [PVBusLogger](#).**cluster0.DAP.mapper**Type: [PVBusMapper](#).**cluster0.DSU**Type: [DSU](#).**cluster0.DSU.PPU_cluster**Type: [PPUv1](#).**cluster0.DSU.PPU_cluster.busslave**Type: [PVBusSlave](#).**cluster0.DSU.PPU_core0**Type: [PPUv1](#).**cluster0.DSU.PPU_core0.busslave**Type: [PVBusSlave](#).**cluster0.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache**

PV Cache.

Type: [PVCache](#).**cluster0.DSU.shared_cache.upstream[0]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[1]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[2]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[3]**Type: [PVBusSlave](#).**cluster0.DSU.shared_cache.upstream[4]**Type: [PVBusSlave](#).**cluster0.DSU.utility_slave[0]**Type: [PVBusSlave](#).**cluster0.MMAP**Type: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Neoverse-V3AE CT model.

Type: [ARM_Neoverse-V3AE](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l2cache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l2cache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l2cache.upstream[1]

Type: PVBusSlave.

cluster0.ext_bus

Type: PVBusLogger.

cluster0.ext_bus.mapper

Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0_labeller

Type: Labeller.

cluster0_labeller.pvbusmodifier

Type: PVBusMapper.

cluster1

ARM Neoverse-V3AE Cluster CT model.

Type: Cluster_ARM_Neoverse-V3AE.

cluster1.AMU

Type: PVBusLogger.

cluster1.AMU.mapper
Type: [PVBusMapper](#).

cluster1.DAP
Type: [PVBusLogger](#).

cluster1.DAP.mapper
Type: [PVBusMapper](#).

cluster1.DSU
Type: [DSU](#).

cluster1.DSU.PPU_cluster
Type: [PPUv1](#).

cluster1.DSU.PPU_cluster.busslave
Type: [PVBusSlave](#).

cluster1.DSU.PPU_core0
Type: [PPUv1](#).

cluster1.DSU.PPU_core0.busslave
Type: [PVBusSlave](#).

cluster1.DSU.mpam_busslave
Type: [PVBusSlave](#).

cluster1.DSU.shared_cache
PV Cache.
Type: [PVCache](#).

cluster1.DSU.shared_cache.upstream[0]
Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[1]
Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[2]
Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[3]
Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[4]
Type: [PVBusSlave](#).

cluster1.DSU.utility_slave[0]
Type: [PVBusSlave](#).

cluster1.MMAP
Type: [PVBusLogger](#).

cluster1.MMAP.mapper
Type: [PVBusMapper](#).

cluster1.cpu0

ARM Neoverse-V3AE CT model.

Type: [ARM_Neoverse-V3AE](#).**cluster1.cpu0.UTLB**

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster1.cpu0.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster1.cpu0.l1icache**

PV Cache.

Type: PVCache.

cluster1.cpu0.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster1.cpu0.l2cache**

PV Cache.

Type: PVCache.

cluster1.cpu0.l2cache.upstream[0]Type: [PVBusSlave](#).**cluster1.cpu0.l2cache.upstream[1]**Type: [PVBusSlave](#).**cluster1.ext_bus**Type: [PVBusLogger](#).**cluster1.ext_bus.mapper**Type: [PVBusMapper](#).**cluster1.gic_cpuif_decoder_cluster**

Type: GICv3CPUInterfaceDecoder.

cluster1_labellerType: [Labeller](#).**cluster1_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: [GICv3Distributor](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

12.72 FVP_Base_Neoverse-V3x1-Neoverse-V3x1

List of instances in FVP_Base_Neoverse-V3x1-Neoverse-V3x1.

FVP_Base_Neoverse-V3x1-Neoverse-V3x1 instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk
ARM Generic Timer.
Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8
Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9
Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase
Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.sm91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

bp.sp810_sysctrl.busslave

Type: `PVBusSlave`.

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.sram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.sram.bus_slave

Type: `PVBusSlave`.

bp.terminal_0

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_1

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_2

Telnet terminal interface.

Type: `TelnetTerminal`.

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [Tzc_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3

Type: [PVBusMapper](#).

bp.ve_sysregs

Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave

Type: [PVBusSlave](#).

bp.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

bp.virtio_net

VirtioNet device over MMIO transport.

Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master

Type: `PVBusMaster`.

bp.virtio_net_labeller

Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: `VirtioEntropyMMIO`.

bp.virtio_rng.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtio_blockdevice.dma_master

Type: `PVBusMaster`.

bp.virtio_blockdevice_labeller

Type: `Labeller`.

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtio_p9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtio_p9device.mmio_slave

Type: `PVBusSlave`.

bp.virtio_p9device.virtio_master

Type: `PVBusMaster`.

bp.virtio_p9device_labeller

Type: `Labeller`.

bp.virtio_p9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Neoverse-V3 Cluster CT model.

Type: [cluster_ARM_Neoverse-V3](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUV1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core0

Type: [PPUV1](#).

cluster0.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache

PV Cache.

Type: [PVCache](#).

cluster0.DSU.shared_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Neoverse-V3 CT model.

Type: [ARM_Neoverse-V3](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapperType: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**Type: [GICv3CPUInterfaceDecoder](#).**cluster0.labeller**Type: [Labeller](#).**cluster0.labeller.pvbusmodifier**Type: [PVBusMapper](#).**cluster1**

ARM Neoverse-V3 Cluster CT model.

Type: [Cluster_ARM_Neoverse-V3](#).**cluster1.AMU**Type: [PVBusLogger](#).**cluster1.AMU.mapper**Type: [PVBusMapper](#).**cluster1.DAP**Type: [PVBusLogger](#).**cluster1.DAP.mapper**Type: [PVBusMapper](#).**cluster1.DSU**Type: [DSU](#).**cluster1.DSU.PPU_cluster**Type: [PPUv1](#).**cluster1.DSU.PPU_cluster.busslave**Type: [PVBusSlave](#).**cluster1.DSU.PPU_core0**Type: [PPUv1](#).**cluster1.DSU.PPU_core0.busslave**Type: [PVBusSlave](#).**cluster1.DSU.mpam_busslave**Type: [PVBusSlave](#).**cluster1.DSU.shared_cache**

PV Cache.

Type: [PVCache](#).**cluster1.DSU.shared_cache.upstream[0]**Type: [PVBusSlave](#).**cluster1.DSU.shared_cache.upstream[1]**Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.DSU.shared_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster1.MMAP

Type: [PVBusLogger](#).

cluster1.MMAP.mapper

Type: [PVBusMapper](#).

cluster1.cpu0

ARM Neoverse-V3 CT model.

Type: [ARM_Neoverse-V3](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster1.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster1.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l2cache

PV Cache.

Type: [PVCache](#).

cluster1.cpu0.l2cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.cpu0.l2cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.ext_bus

Type: [PVBusLogger](#).

cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

cluster1.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster1_labeller

Type: [Labeller](#).

cluster1_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_0_1

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_1_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_tl

Type: `GICv3Distributor`.

pctl

Base Platforms Power Controller.

Type: `Base_PowerController`.

pctl.busslave

Type: `PVBusSlave`.

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

12.73 FVP_Base_RevC-2xAEMvA

List of instances in FVP_Base_RevC-2xAEMvA.

FVP_Base_RevC-2xAEMvA instances

address_map_terminator

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5Type: [PVBUSSlave](#).**bp.ap_refclk.busbase6**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase7**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase8**Type: [PVBUSSlave](#).**bp.ap_refclk.busbase9**Type: [PVBUSSlave](#).**bp.ap_refclk.busctlbase**Type: [PVBUSSlave](#).**bp.audioout**

SDL based Audio Output for PL041_AACI.

Type: [AudioOut SDL](#).**bp.clock100Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.clock24MHz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.clock300MHz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.clock32KHz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**bp.clock35MHz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBusSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.psram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.psram.bus_slave

Type: `PVBusSlave`.

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

bp.refcounter.pvbus_control_s[0]

Type: `PVBusSlave`.

bp.refcounter.pvbus_read_s[0]

Type: `PVBusSlave`.

bp.reset_or

Or Gate.

Type: `OrGate`.

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rl_dram.bus_slave

Type: `PVBusSlave`.

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.rt_dram.bus_slave

Type: `PVBusSlave`.

bp.s_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.s_dram.bus_slave

Type: `PVBusSlave`.

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBUSSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBUSSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBUSMapper](#).

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier
Type: `PVBusMapper`.

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: `virtioEntropyMMIO`.

bp.virtio_rng.dma_master
Type: `PVBusMaster`.

bp.virtioblockdevice

VirtioBlock device over MMIO transport.

Type: `virtioBlockMMIO`.

bp.virtioblockdevice.dma_master

Type: `PVBusMaster`.

bp.virtioblockdevice_labeller

Type: `Labeller`.

bp.virtioblockdevice_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.virtiop9device

virtio P9 server.

Type: `VirtioP9Device`.

bp.virtiop9device.mmio_slave

Type: `PVBusSlave`.

bp.virtiop9device.virtio_master

Type: `PVBusMaster`.

bp.virtiop9device_labeller

Type: `Labeller`.

bp.virtiop9device_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci550

This is a cache coherent interconnect.

Type: [CCI550](#).

cci550.pvbus_register_file_s[0]

Type: [PVBusSlave](#).

cci550.upstream[0]

Type: [PVBusSlave](#).

cci550.upstream[1]

Type: [PVBusSlave](#).

cci550.upstream[2]

Type: [PVBusSlave](#).

cci550.upstream[3]

Type: [PVBusSlave](#).

cci550.upstream[4]

Type: [PVBusSlave](#).

cci550.upstream[5]

Type: [PVBusSlave](#).

cci550.upstream[6]

Type: [PVBusSlave](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM AEMvA Cluster CT model.

Type: [Cluster_ARM_AEM-A_MP](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.mpam_busslave

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: pVCache.

cluster0.cpu0.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: pVCache.

cluster0.cpu0.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu1**

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).**cluster0.cpu1.UTLB**

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: pVCache.

cluster0.cpu1.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster0.cpu1.l1icache**

PV Cache.

Type: pVCache.

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).

cluster0.cpu2.UTLB

Type: TLB.

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).

cluster0.cpu3.UTLB

Type: TLB.

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu3.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: `PVCache`.

`cluster0.cpu3.l1icache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.ext_bus`
Type: `PVBusLogger`.

`cluster0.ext_bus.mapper`
Type: `PVBusMapper`.

`cluster0.gic_cpuif_decoder_cluster`
Type: `GICv3CPUInterfaceDecoder`.

`cluster0.l2_cache`
PV Cache.
Type: `PVCache`.

`cluster0.l2_cache.upstream[0]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[10]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[11]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[12]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[13]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[14]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[15]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[16]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[1]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[2]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[3]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[4]`
Type: `PVBusSlave`.

`cluster0.l2_cache.upstream[5]`
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[6]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]
Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

cluster0.l2_flusher
Type: [AsyncCacheFlushUnit](#).

cluster0_labeller
Type: [Labeller](#).

cluster0_labeller.pvbusmodifier
Type: [PVBusMapper](#).

cluster1
ARM AEMvA Cluster CT model.
Type: [Cluster_ARM_AEM-A_MP](#).

cluster1.AMU
Type: [PVBusLogger](#).

cluster1.AMU.mapper
Type: [PVBusMapper](#).

cluster1.DAP
Type: [PVBusLogger](#).

cluster1.DAP.mapper
Type: [PVBusMapper](#).

cluster1.DSU
Type: [DSU](#).

cluster1.DSU.mpam_busslave
Type: [PVBusSlave](#).

cluster1.MMAP
Type: [PVBusLogger](#).

cluster1.MMAP.mapper
Type: [PVBusMapper](#).

cluster1.acp_mapper
Type: [PVBusMapper](#).

cluster1.cpu0
ARM AEM-A MP CT model.
Type: [ARM_AEM-A_MP](#).

cluster1.cpu0.UTLB

Type: TLB.

cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1

ARM AEM-A MP CT model.

Type: ARM_AEM-A_MP.

cluster1.cpu1.UTLB

Type: TLB.

cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu1.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster1.cpu1.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster1.cpu2

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).

cluster1.cpu2.UTLB

Type: TLB.

cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu2.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu2.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu2.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.cpu2.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu2.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.cpu3

ARM AEM-A MP CT model.

Type: [ARM_AEM-A_MP](#).

cluster1.cpu3.UTLB

Type: TLB.

cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: Tlbcadi.

cluster1.cpu3.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster1.cpu3.l1dcache

PV Cache.

Type: Pvcache.

cluster1.cpu3.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.cpu3.l1icache

PV Cache.

Type: Pvcache.

cluster1.cpu3.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster1.ext_bus

Type: [PVBusLogger](#).

cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

cluster1.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster1.l2_cache.upstream[7]Type: [PVBUSSlave](#).**cluster1.l2_cache.upstream[8]**Type: [PVBUSSlave](#).**cluster1.l2_cache.upstream[9]**Type: [PVBUSSlave](#).**cluster1.l2_flusher**Type: [AsyncCacheFlushUnit](#).**cluster1_labeller**Type: [Labeller](#).**cluster1_labeller.pvbusmodifier**Type: [PVBUSMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBUSLogger](#).**dapmemlogger.mapper**Type: [PVBUSMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBUSMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.ITS0**Type: [GICv3InterruptTranslationService](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_1_0

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_1_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_1_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_1_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_1_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_1_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_1_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_1_3_0

Type: GICv3Redistributor.

gic_distributor.rd_t1

Type: GICv3Distributor.

gpu

G76 addon for the Base Platform.

Type: G76_DB.

gpu.gpu

ARM Mali-G76 GPU.

Type: [Mali_G76](#).

gpu.gpu.busmaster

Type: [PVBusMaster](#).

gpu.gpu.busslave

Type: [PVBusSlave](#).

pci

PCI addon for the Base Platform.

Type: [BasePlatformPCISBSA](#).

pci.dma330x4

Type: [DMA330x4](#).

pci.dma330x4.decoder

Type: [PVBusMapper](#).

pci.dma330x4.dmac0

ARM PrimeCell DMA Controller(PL330).

Type: [PL330_DMACH](#).

pci.dma330x4.dmac0.busmaster

Type: [PVBusMaster](#).

pci.dma330x4.dmac0.busslave

Type: [PVBusSlave](#).

pci.dma330x4.dmac0.busslave_ns

Type: [PVBusSlave](#).

pci.dma330x4.dmac0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pci.dma330x4.dmac0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pci.dma330x4.dmac0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pci.dma330x4.dmac0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pci.dma330x4.dmac1

ARM PrimeCell DMA Controller(PL330).

Type: [PL330_DMACH](#).

pci.dma330x4.dmac1.busmaster

Type: [PVBUSMASTER](#).

pci.dma330x4.dmac1.busslave

Type: [PVBUSSLAVE](#).

pci.dma330x4.dmac1.busslave_ns

Type: [PVBUSSLAVE](#).

pci.dma330x4.dmac1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pci.dma330x4.dmac1.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pci.dma330x4.dmac1.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pci.dma330x4.dmac1.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pci.dma330x4.dmac2

ARM PrimeCell DMA Controller(PL330).

Type: [PL330_DMACH](#).

pci.dma330x4.dmac2.busmaster

Type: [PVBUSMASTER](#).

pci.dma330x4.dmac2.busslave

Type: [PVBUSSLAVE](#).

pci.dma330x4.dmac2.busslave_ns

Type: [PVBUSSLAVE](#).

pci.dma330x4.dmac2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pci.dma330x4.dmac2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pci.dma330x4.dmac2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pci.dma330x4.dmac2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pci.dma330x4.dmac3

ARM PrimeCell DMA Controller(PL330).

Type: [PL330_DMAC](#).

pci.dma330x4.dmac3.busmaster

Type: [PVBusMaster](#).

pci.dma330x4.dmac3.busslave

Type: [PVBusSlave](#).

pci.dma330x4.dmac3.busslave_ns

Type: [PVBusSlave](#).

pci.dma330x4.dmac3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pci.dma330x4.dmac3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pci.dma330x4.dmac3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pci.dma330x4.dmac3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pci.dma330x4.label0

Type: [PVBusMapper](#).

pci.dma330x4.label1

Type: [PVBusMapper](#).

pci.dma330x4.label2

Type: [PVBusMapper](#).

pci.dma330x4.label3

Type: [PVBusMapper](#).

pci.dma330x4.label_smmuv3testengine

Type: [PVBusMapper](#).

pci.dma330x4.smmuv3testengine

SMMUv3 Test Engine.

Type: [SMMUv3TestEngine](#).

pci.dma330x4.smmuv3testengine.register_file[0]

Type: [PVBusSlave](#).

pci.pci_smmuv3

System MMUv3 configured for PCI-E Sub-system.

Type: [SMMUv3_FOR_PCIE](#).

pci.pci_smmuv3.mmu

SMMUv3 AEM.

Type: [SMMUv3AEM](#).

pci.pci_smmuv3.mmu.register_file[0]

Type: [PVBusSlave](#).

pci.pci_smmuv3.mmu.service_request_tbu[0]

Type: [PVBusSlave](#).

`pci.pci_smmuv3.mmu.service_request_tbu[10]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[11]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[12]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[13]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[14]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[15]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[16]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[17]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[18]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[19]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[1]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[20]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[21]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[22]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[23]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[24]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[25]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[26]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[27]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[28]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[29]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[2]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[30]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[31]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[32]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[33]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[34]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[35]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[36]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[37]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[38]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[39]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[3]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[40]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[41]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[42]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[43]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[44]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[45]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[46]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[47]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[48]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[49]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[4]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[50]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[51]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[52]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[53]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[54]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[55]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[56]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[57]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[58]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[59]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[5]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[60]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[61]`
Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[62]`

Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[63]`

Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[6]`

Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[7]`

Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[8]`

Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.service_request_tbu[9]`

Type: `PVBusSlave`.

`pci.pci_smmuv3.mmu.tbu[0]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[10]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[11]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[12]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[13]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[14]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[15]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[16]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[17]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[18]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[19]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[1]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[20]`

Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[21]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[22]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[23]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[24]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[25]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[26]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[27]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[28]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[29]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[2]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[30]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[31]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[32]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[33]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[34]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[35]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[36]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[37]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[38]`
Type: `PVBusMapper`.

`pci.pci_smmuv3.mmu.tbu[39]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[3]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[40]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[41]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[42]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[43]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[44]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[45]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[46]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[47]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[48]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[49]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[4]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[50]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[51]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[52]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[53]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[54]`

Type: [PVBusMapper](#).

`pci.pci_smmuv3.mmu.tbu[55]`

Type: [PVBusMapper](#).

pci.pci_smmuv3.mmu.tbu[56]Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[57]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[58]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[59]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[5]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[60]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[61]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[62]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[63]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[6]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[7]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[8]**Type: [PVBusMapper](#).**pci.pci_smmuv3.mmu.tbu[9]**Type: [PVBusMapper](#).**pci.pci_smmuv3_msirewriter**

Recognise writes to the GITS_TRANSLATER register and rewrite them to go to the GITS_TRANSLATE64R register. The DeviceID is expected to be in the bottom 32 bits of ExtendedID of the transaction. Debug transactions and reads are not rewritten. This component can also, optionally, apply a 16 bit 'label' to the top 16 bits of the MasterID in the same way that the Labeller component does.

Type: [MSIRewriter](#).**pci.pci_smmuv3_msirewriter.mapper**Type: [PVBusMapper](#).**pci.pci_smmuv3_msirewriter.pvbusmaster**Type: [PVBusMaster](#).**pci.pci_smmuv3_msirewriter.pvbusslave**Type: [PVBusSlave](#).

pci.pcie_rc
PCleRootComplex.
Type: PCIeRootComplex.

pci.pcie_rc.__downstream__0
Type: transaction_router_tracer.

pci.pcie_rc.__downstream__0.cfglogger
Type: PVBusLogger.

pci.pcie_rc.__downstream__0.cfglogger.mapper
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.config_mapper
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.config_slave
Type: PVBusSlave.

pci.pcie_rc.__downstream__0.device_mapper
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.devicelogger
Type: PVBusLogger.

pci.pcie_rc.__downstream__0.devicelogger.mapper
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.dma_connector
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.dmalogger
Type: PVBusLogger.

pci.pcie_rc.__downstream__0.dmalogger.mapper
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.id_routed_mapper
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.mem_connector
Type: PVBusMapper.

pci.pcie_rc.__downstream__0.probe_master
Type: PVBusMaster.

pci.pcie_rc.__downstream__0.unsupported_request_slave
Type: PVBusSlave.

pci.pcie_rc.__downstream__0.unsupported_request_slave_raowi
Type: PVBusSlave.

pci.pcie_rc.__downstream__1
Type: transaction_router_tracer.

`pci.pcie_rc.__downstream__1.cfglogger`
Type: `PVBusLogger`.

`pci.pcie_rc.__downstream__1.cfglogger.mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.config_mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.config_slave`
Type: `PVBusSlave`.

`pci.pcie_rc.__downstream__1.device_mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.devicellogger`
Type: `PVBusLogger`.

`pci.pcie_rc.__downstream__1.devicellogger.mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.dma_connector`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.dmalogger`
Type: `PVBusLogger`.

`pci.pcie_rc.__downstream__1.dmalogger.mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.id_routed_mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.mem_connector`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__1.probe_master`
Type: `PVBusMaster`.

`pci.pcie_rc.__downstream__1.unsupported_request_slave`
Type: `PVBusSlave`.

`pci.pcie_rc.__downstream__1.unsupported_request_slave_raowi`
Type: `PVBusSlave`.

`pci.pcie_rc.__downstream__2`
Type: `transaction_router_tracer`.

`pci.pcie_rc.__downstream__2.cfglogger`
Type: `PVBusLogger`.

`pci.pcie_rc.__downstream__2.cfglogger.mapper`
Type: `PVBusMapper`.

`pci.pcie_rc.__downstream__2.config_mapper`
Type: `PVBusMapper`.

pci.pcie_rc.__downstream__2.config_slave
Type: [PVBUSSlave](#).

pci.pcie_rc.__downstream__2.device_mapper
Type: [PVBUSMapper](#).

pci.pcie_rc.__downstream__2.devicelogger
Type: [PVBUSLogger](#).

pci.pcie_rc.__downstream__2.devicelogger.mapper
Type: [PVBUSMapper](#).

pci.pcie_rc.__downstream__2.dma_connector
Type: [PVBUSMapper](#).

pci.pcie_rc.__downstream__2.dmalogger
Type: [PVBUSLogger](#).

pci.pcie_rc.__downstream__2.dmalogger.mapper
Type: [PVBUSMapper](#).

pci.pcie_rc.__downstream__2.id_routed_mapper
Type: [PVBUSMapper](#).

pci.pcie_rc.__downstream__2.mem_connector
Type: [PVBUSMapper](#).

pci.pcie_rc.__downstream__2.probe_master
Type: [PVBUSMaster](#).

pci.pcie_rc.__downstream__2.unsupported_request_slave
Type: [PVBUSSlave](#).

pci.pcie_rc.__downstream__2.unsupported_request_slave_raowi
Type: [PVBUSSlave](#).

pci.pcie_rc.achil
Type: [achil](#).

pci.pcie_rc.achil.ahci_sata
Type: [ahci_sata](#).

pci.pcie_rc.achil.ahci_sata.ahci_master
Type: [PVBUSMaster](#).

pci.pcie_rc.achil.ahci_sata.register_slave
Type: [PVBUSSlave](#).

pci.pcie_rc.achil.endpoint
Type: [pcie_endpoint_tracer](#).

pci.pcie_rc.achil.endpoint.config_slave
Type: [PVBUSSlave](#).

pci.pcie_rc.achil.endpoint.dma_connector
Type: [PVBUSMapper](#).

pci.pcie_rc.achi1.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.achi1.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.achi1.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.achi2
Type: [achi2](#).

pci.pcie_rc.achi2.ahci_sata
Type: [ahci_sata](#).

pci.pcie_rc.achi2.ahci_sata.ahci_master
Type: [PVBusMaster](#).

pci.pcie_rc.achi2.ahci_sata.register_slave
Type: [PVBusSlave](#).

pci.pcie_rc.achi2.endpoint
Type: [pcie_endpoint_tracer](#).

pci.pcie_rc.achi2.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.achi2.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.achi2.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.achi2.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.achi2.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.ahci0
Type: [ahci0](#).

pci.pcie_rc.ahci0.ahci_sata
Type: [ahci_sata](#).

pci.pcie_rc.ahci0.ahci_sata.ahci_master
Type: [PVBusMaster](#).

pci.pcie_rc.ahci0.ahci_sata.register_slave
Type: [PVBusSlave](#).

pci.pcie_rc.ahci0.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.ahci0.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.ahci0.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.ahci0.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.ahci0.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.ahci0.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.hostbridge0
Type: [hostbridge0](#).

pci.pcie_rc.hostbridge0.endpoint
Type: [pcie_endpoint_tracer](#).

pci.pcie_rc.hostbridge0.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.hostbridge0.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.hostbridge0.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.hostbridge0.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.hostbridge0.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.hostbridge0.ignore_slave
Type: [PVBusSlave](#).

pci.pcie_rc.pvbus2pci0
Type: [transaction_router_tracer](#).

pci.pcie_rc.pvbus2pci0.cfglogger
Type: [PVBusLogger](#).

pci.pcie_rc.pvbus2pci0.cfglogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.config_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.device_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.devicellogger
Type: [PVBusLogger](#).

pci.pcie_rc.pvbus2pci0.devicelogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.dma_attr_modifier
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.dmalogger
Type: [PVBusLogger](#).

pci.pcie_rc.pvbus2pci0.dmalogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.id_routed_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci0.probe_master
Type: [PVBusMaster](#).

pci.pcie_rc.pvbus2pci0.rootcomplex_error_message_receiver[0]
Type: [PVBusSlave](#).

pci.pcie_rc.pvbus2pci0_logger
Type: [PVBusLogger](#).

pci.pcie_rc.pvbus2pci0_logger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.pvbus2pci_idmapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0
PCleBridge.
Type: [PCleBridge](#).

pci.pcie_rc.rootport0.cfglogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport0.cfglogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.config_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport0.device_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.devicelogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport0.devicelogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.dmalogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport0.dmalogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.id_routed_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport0.probe_master
Type: [PVBusMaster](#).

pci.pcie_rc.rootport0.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport0.unsupported_request_slave_raowi
Type: [PVBusSlave](#).

pci.pcie_rc.rootport1
PCleBridge.
Type: [PCIEBridge](#).

pci.pcie_rc.rootport1.cfglogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport1.cfglogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.config_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport1.device_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.devicelogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport1.devicelogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.dmalogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport1.dmalogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.id_routed_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport1.probe_master
Type: [PVBusMaster](#).

pci.pcie_rc.rootport1.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport1.unsupported_request_slave_raowi
Type: [PVBusSlave](#).

pci.pcie_rc.rootport2
PCleBridge.
Type: [PCIEBridge](#).

pci.pcie_rc.rootport2.cfglogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport2.cfglogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.config_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport2.device_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.devicelogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport2.devicelogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.dmalogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport2.dmalogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.id_routed_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport2.probe_master
Type: [PVBusMaster](#).

pci.pcie_rc.rootport2.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport2.unsupported_request_slave_raowi
Type: [PVBusSlave](#).

pci.pcie_rc.rootport3
PCleBridge.
Type: [PCIEBridge](#).

pci.pcie_rc.rootport3.cfglogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport3.cfglogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.config_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport3.device_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.devicelogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport3.devicelogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.dmalogger
Type: [PVBusLogger](#).

pci.pcie_rc.rootport3.dmalogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.id_routed_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.rootport3.probe_master
Type: [PVBusMaster](#).

pci.pcie_rc.rootport3.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.rootport3.unsupported_request_slave_raowi
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine0
Type: [smmuv3testengine0](#).

pci.pcie_rc.smmuv3testengine0.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine0.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine0.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine0.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine0.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine0.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine0.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine0.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine1
Type: [smmuv3testengine1](#).

pci.pcie_rc.smmuv3testengine1.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine1.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine1.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine1.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine1.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine1.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine1.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine1.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine2
Type: [smmuv3testengine2](#).

pci.pcie_rc.smmuv3testengine2.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine2.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine2.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine2.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine2.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine2.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine2.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine2.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine3
Type: [smmuv3testengine3](#).

pci.pcie_rc.smmuv3testengine3.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine3.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine3.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine3.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine3.endpoint.pvbus_id_routed_s[0]

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine3.endpoint.unsupported_request_slave

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine3.smmuv3testengine

SMMUv3 Test Engine.

Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine3.smmuv3testengine.register_file[0]

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine4

Type: [smmuv3testengine4](#).

pci.pcie_rc.smmuv3testengine4.endpoint

PCIEEndpoint.

Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine4.endpoint.config_slave

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine4.endpoint.dma_connector

Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine4.endpoint.mem_connector

Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine4.endpoint.pvbus_id_routed_s[0]

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine4.endpoint.unsupported_request_slave

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine4.smmuv3testengine

SMMUv3 Test Engine.

Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine4.smmuv3testengine.register_file[0]

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine5

Type: [smmuv3testengine5](#).

pci.pcie_rc.smmuv3testengine5.endpoint

PCIEEndpoint.

Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine5.endpoint.config_slave

Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine5.endpoint.dma_connector

Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine5.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine5.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine5.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine5.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine5.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine6
Type: [smmuv3testengine6](#).

pci.pcie_rc.smmuv3testengine6.endpoint
PCleEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine6.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine6.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine6.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine6.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine6.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine6.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine6.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine7
Type: [smmuv3testengine7](#).

pci.pcie_rc.smmuv3testengine7.endpoint
PCleEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine7.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine7.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine7.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine7.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine7.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine7.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine7.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine8
Type: [smmuv3testengine8](#).

pci.pcie_rc.smmuv3testengine8.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine8.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine8.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine8.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine8.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine8.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine8.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine8.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine9
Type: [smmuv3testengine9](#).

pci.pcie_rc.smmuv3testengine9.endpoint
PCIEEndpoint.
Type: [PCIEEndpoint](#).

pci.pcie_rc.smmuv3testengine9.endpoint.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine9.endpoint.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine9.endpoint.mem_connector
Type: [PVBusMapper](#).

pci.pcie_rc.smmuv3testengine9.endpoint.pvbus_id_routed_s[0]
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine9.endpoint.unsupported_request_slave
Type: [PVBusSlave](#).

pci.pcie_rc.smmuv3testengine9.smmuv3testengine
SMMUv3 Test Engine.
Type: [SMMUv3TestEngine](#).

pci.pcie_rc.smmuv3testengine9.smmuv3testengine.register_file[0]
Type: [PVBusSlave](#).

pci.pcie_rc.switch0.upstream
Type: [transaction_router_tracer](#).

pci.pcie_rc.switch0.upstream.cfglogger
Type: [PVBusLogger](#).

pci.pcie_rc.switch0.upstream.cfglogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.config_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.config_slave
Type: [PVBusSlave](#).

pci.pcie_rc.switch0.upstream.device_mapper
Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.devicellogger
Type: [PVBusLogger](#).

pci.pcie_rc.switch0.upstream.devicellogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.dma_connector
Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.dmalogger
Type: [PVBusLogger](#).

pci.pcie_rc.switch0.upstream.dmalogger.mapper
Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.id_routed_mapper

Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.mem_connector

Type: [PVBusMapper](#).

pci.pcie_rc.switch0.upstream.probe_master

Type: [PVBusMaster](#).

pci.pcie_rc.switch0.upstream.unsupported_request_slave

Type: [PVBusSlave](#).

pci.pcie_rc.switch0.upstream.unsupported_request_slave_raowi

Type: [PVBusSlave](#).

pci.smmulogger

Bus Logger.

Type: [PVBusLogger](#).

pci.smmulogger.mapper

Type: [PVBusMapper](#).

pci.tbu0_pre_smmu_logger

Bus Logger.

Type: [PVBusLogger](#).

pci.tbu0_pre_smmu_logger.mapper

Type: [PVBusMapper](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

test_engine

Test Engine for base platform.

Type: [TestEngine](#).

test_engine.device0

SMMUv3 Test Engine.

Type: [SMMUv3TestEngine](#).

test_engine.device0.register_file[0]

Type: [PVBusSlave](#).

13. BaseR Platform FVPs

This chapter describes the BaseR Platform FVPs contained in the FVP Standard Library.

For the BaseR Platform memory map, see [BaseR Platform memory map](#) in the *Fast Models Reference Guide*.

13.1 FVP_BaseR_Cortex-R52Plus

List of instances in FVP_BaseR_Cortex-R52Plus.

FVP_BaseR_Cortex-R52Plus instances

address_map_terminator

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.hdlcd0.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.hdlcd0.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.hdlcd0_labeller

Type: `Labeller`.

bp.hdlcd0_labeller.pvbusmodifier

Type: `PVBusMapper`.

bp.hostbridge

Host Socket Interface Component.

Type: `HostBridge`.

bp.lcd_security_map

Type: `PVBusMapper`.

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: `LS64TestingFIFO`.

bp.ls64_testing_fifo.pvbusslave

Type: `PVBusSlave`.

bp.mmc

Generic Multimedia Card.

Type: `MMC`.

bp.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

bp.mpe.mapper

Type: `PVBusMapper`.

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

bp.nontrustedrom.map

Type: `PVBusMapper`.

bp.nontrustedrom.mbs

Type: `PVBusSlave`.

bp.nontrustedrom.rmbs

Type: `PVBusSlave`.

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

bp.ns_dram.bus_slave

Type: `PVBusSlave`.

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-R52Plus Cluster CT model.

Type: [Cluster_ARM_Cortex-R52Plus](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.MMAPType: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM Cortex-R52Plus CT model.

Type: [ARM_Cortex-R52Plus](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu1**

ARM Cortex-R52Plus CT model.

Type: [ARM_Cortex-R52Plus](#).**cluster0.cpu1.UTLB**

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu1.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu1.l1dcache**

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-R52Plus CT model.

Type: [ARM_Cortex-R52Plus](#).

cluster0.cpu2.UTLB

Type: [TLB](#).

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-R52Plus CT model.

Type: [ARM_Cortex-R52Plus](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.gic_iri
GIC Interrupt Redistribution Infrastructure component.
Type: `GIC_IRI`.

cluster0.gic_iri.rd_0
Type: `GICv3RedistributorInternal`.

cluster0.gic_iri.rd_0_0
Type: `GICv3RedistributorInternal`.

cluster0.gic_iri.rd_0_0_0
Type: `GICv3RedistributorInternal`.

cluster0.gic_iri.rd_0_0_0_0
Type: `GICv3Redistributor`.

cluster0.gic_iri.rd_0_0_0_1
Type: `GICv3Redistributor`.

cluster0.gic_iri.rd_0_0_0_2
Type: `GICv3Redistributor`.

cluster0.gic_iri.rd_0_0_0_3
Type: `GICv3Redistributor`.

cluster0.gic_iri.rd_0_0_0_4
Type: `GICv3Redistributor`.

cluster0.gic_iri.rd_t1
Type: `GICv3Distributor`.

cluster0.l2_flusher
Type: `AsyncCacheFlushUnit`.

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

flash_ram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

flash_ram0.bus_slave

Type: [PVBusSlave](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

13.2 FVP_BaseR_Cortex-R52x1

List of instances in FVP_BaseR_Cortex-R52x1.

FVP_BaseR_Cortex-R52x1 instances

address_map_terminator

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBUSMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbsType: [PVBusSlave](#).**bp.flash0.rmbs**Type: [PVBusSlave](#).**bp.flash1**

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).**bp.flash1.map**Type: [PVBusMapper](#).**bp.flash1.mbs**Type: [PVBusSlave](#).**bp.flash1.rmbs**Type: [PVBusSlave](#).**bp.flashloader0**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.flashloader1**

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).**bp.generic_watchdog**

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).**bp.generic_watchdog.busctlbase**Type: [PVBusSlave](#).**bp.generic_watchdog.busrefbase**Type: [PVBusSlave](#).**bp.hdlcd0**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**bp.hdlcd0.busmaster**Type: [PVBusMaster](#).**bp.hdlcd0.busslave**Type: [PVBusSlave](#).**bp.hdlcd0.timer**

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBusSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBusSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl1111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.r1_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.secureflash.map
Type: [PVBusMapper](#).

bp.secureflash.mbs
Type: [PVBusSlave](#).

bp.secureflash.rmbs
Type: [PVBusSlave](#).

bp.secureflashloader
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.smsc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

bp.smsc_91c111.SMSC slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM CortexR52 MP Cluster CT model.

Type: [Cluster_ARM_Cortex-R52](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.MMAPType: [PVBusLogger](#).**cluster0.MMAP.mapper**Type: [PVBusMapper](#).**cluster0.acp_mapper**Type: [PVBusMapper](#).**cluster0.cpu0**

ARM CortexR52 MP CT model.

Type: [ARM_CortexR52](#).**cluster0.cpu0.UTLB**

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster0.cpu0.gicv3_cpu_if**Type: [GICv3CPUInterface](#).**cluster0.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster0.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster0.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster0.ext_bus**Type: [PVBusLogger](#).**cluster0.ext_bus.mapper**Type: [PVBusMapper](#).**cluster0.gic_cpuif_decoder_cluster**Type: [GICv3CPUInterfaceDecoder](#).**cluster0.gic_iri**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**cluster0.gic_iri.rd_0**Type: [GICv3RedistributorInternal](#).**cluster0.gic_iri.rd_0_0**Type: [GICv3RedistributorInternal](#).

cluster0.gic_iri.rd_0_0_0

Type: GICv3RedistributorInternal.

cluster0.gic_iri.rd_0_0_0_0

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_1

Type: GICv3Redistributor.

cluster0.gic_iri.rd_tl

Type: GICv3Distributor.

cluster0.l2_flusher

Type: AsyncCacheFlushUnit.

cluster0_labeller

Type: Labeller.

cluster0_labeller.pvbusmodifier

Type: PVBusMapper.

dapmemlogger

Bus Logger.

Type: PVBusLogger.

dapmemlogger.mapper

Type: PVBusMapper.

elfloader

ELF loader component.

Type: ElfLoader.

elfloader.pvbus_busmaster

Type: PVBusMaster.

flash_ram0

RAM device, can be dynamic or static ram.

Type: RAMDevice.

flash_ram0.bus_slave

Type: PVBusSlave.

pctl

Base Platforms Power Controller.

Type: Base_PowerController.

pctl.busslave

Type: PVBusSlave.

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

13.3 FVP_BaseR_Cortex-R52x2

List of instances in FVP_BaseR_Cortex-R52x2.

FVP_BaseR_Cortex-R52x2 instances

address_map_terminator

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBUSSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave
Type: [PVBusSlave](#).

bp.refcounter
Memory Mapped Counter Module for Generic Timers.
Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]
Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]
Type: [PVBusSlave](#).

bp.reset_or
Or Gate.
Type: [OrGate](#).

bp.rl_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`bp.trusted_watchdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`bp.trusted_watchdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`bp.tzc_400`

TrustZone Address Space Controller.

Type: [TZC_400](#).

`bp.tzc_400.apbslave[0]`

Type: [PVBusSlave](#).

`bp.tzc_400.filter0`

Type: `filter0`.

`bp.tzc_400.filter0.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter1`

Type: `filter1`.

`bp.tzc_400.filter1.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter2`

Type: `filter2`.

`bp.tzc_400.filter2.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter3`

Type: `filter3`.

`bp.tzc_400.filter3.BusMapper`

Type: [PVBusMapper](#).

`bp.utility_bus_map0`

Type: [PVBusMapper](#).

`bp.utility_bus_map1`

Type: [PVBusMapper](#).

`bp.utility_bus_map2`

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM CortexR52 MP Cluster CT model.

Type: [Cluster_ARM_Cortex-R52](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM CortexR52 MP CT model.

Type: [ARM_CortexR52](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM CortexR52 MP CT model.

Type: `ARM_CortexR52`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.ext_bus**Type: `PVBusLogger`.**cluster0.ext_bus.mapper**Type: `PVBusMapper`.**cluster0.gic_cpuif_decoder_cluster**Type: `GICv3CPUInterfaceDecoder`.**cluster0.gic_iri**

GIC Interrupt Redistribution Infrastructure component.

Type: `GIC_IRI`.**cluster0.gic_iri.rd_0**Type: `GICv3RedistributorInternal`.**cluster0.gic_iri.rd_0_0**Type: `GICv3RedistributorInternal`.

cluster0.gic_iri.rd_0_0_0

Type: GICv3RedistributorInternal.

cluster0.gic_iri.rd_0_0_0_0

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_1

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_2

Type: GICv3Redistributor.

cluster0.gic_iri.rd_tl

Type: GICv3Distributor.

cluster0.l2_flusher

Type: AsyncCacheFlushUnit.

cluster0_labellerType: [Labeller](#).**cluster0_labeller.pvbusmodifier**Type: [PVBusMapper](#).**dapmemlogger**

Bus Logger.

Type: [PVBusLogger](#).**dapmemlogger.mapper**Type: [PVBusMapper](#).**elfloader**

ELF loader component.

Type: [ElfLoader](#).**elfloader.pvbus_busmaster**Type: [PVBusMaster](#).**flash_ram0**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**flash_ram0.bus_slave**Type: [PVBusSlave](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

13.4 FVP_BaseR_Cortex-R52x4

List of instances in `FVP_BaseR_Cortex-R52x4`.

FVP_BaseR_Cortex-R52x4 instances

address_map_terminator

Type: `PVBusMapper`.

address_swapper

Type: `PVBusMapper`.

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_0_1.busslave

Type: `PVBusSlave`.

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_2_3.busslave

Type: `PVBusSlave`.

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map
Type: [PVBusMapper](#).

bp.flash1.mbs
Type: [PVBusSlave](#).

bp.flash1.rmbs
Type: [PVBusSlave](#).

bp.flashloader0
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.flashloader1
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.generic_watchdog
ARM Generic Watchdog.
Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase
Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave
Type: [PVBusSlave](#).

bp.refcounter
Memory Mapped Counter Module for Generic Timers.
Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]
Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]
Type: [PVBusSlave](#).

bp.reset_or
Or Gate.
Type: [OrGate](#).

bp.rl_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0

Type: [filter0](#).

bp.tzc_400.filter0.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter1

Type: [filter1](#).

bp.tzc_400.filter1.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter2

Type: [filter2](#).

bp.tzc_400.filter2.BusMapper

Type: [PVBusMapper](#).

bp.tzc_400.filter3

Type: [filter3](#).

bp.tzc_400.filter3.BusMapper

Type: [PVBusMapper](#).

bp.utility_bus_map0

Type: [PVBusMapper](#).

bp.utility_bus_map1

Type: [PVBusMapper](#).

bp.utility_bus_map2

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM CortexR52 MP Cluster CT model.

Type: [cluster_ARM_Cortex-R52](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.acp_mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM CortexR52 MP CT model.

Type: [ARM_CortexR52](#).

cluster0.cpu0.UTLB

Type: [TLB](#).

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.**cluster0.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1**

ARM CortexR52 MP CT model.

Type: `ARM_CortexR52`.**cluster0.cpu1.UTLB**Type: `TLB`.**cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu1.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu2**

ARM CortexR52 MP CT model.

Type: `ARM_CortexR52`.**cluster0.cpu2.UTLB**Type: `TLB`.**cluster0.cpu2.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu2.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu2.l1dcache**

PV Cache.

Type: `PVCache`.

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM CortexR52 MP CT model.

Type: [ARM_CortexR52](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.gic_iri

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

cluster0.gic_iri.rd_0

Type: [GICv3RedistributorInternal](#).

cluster0.gic_iri.rd_0_0

Type: GICv3RedistributorInternal.

cluster0.gic_iri.rd_0_0_0

Type: GICv3RedistributorInternal.

cluster0.gic_iri.rd_0_0_0_0

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_1

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_2

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_3

Type: GICv3Redistributor.

cluster0.gic_iri.rd_0_0_0_4

Type: GICv3Redistributor.

cluster0.gic_iri.rd_tl

Type: GICv3Distributor.

cluster0.l2_flusher

Type: AsyncCacheFlushUnit.

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

flash_ram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

flash_ram0.bus_slave

Type: [PVBusSlave](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

13.5 FVP_BaseR_Cortex-R82

List of instances in FVP_BaseR_Cortex-R82.

FVP_BaseR_Cortex-R82 instances

address_map_terminator

Type: [PVBusMapper](#).

address_remap_switch

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

address_remap_switch.pvbus_mapper

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave

Type: [PVBusSlave](#).

bp.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

bp.fixed_security_map

Type: [PVBusMapper](#).

bp.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash0.map

Type: [PVBusMapper](#).

bp.flash0.mbs

Type: [PVBusSlave](#).

bp.flash0.rmbs

Type: [PVBusSlave](#).

bp.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.ns_dram.bus_slave`

Type: `PVBusSlave`.

`bp.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`bp.pl011_uart0.busslave`

Type: `PVBusSlave`.

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TZC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: filter0.

bp.tzc_400.filter0.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter1
Type: filter1.

bp.tzc_400.filter1.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter2
Type: filter2.

bp.tzc_400.filter2.BusMapper
Type: [PVBusMapper](#).

bp.tzc_400.filter3
Type: filter3.

bp.tzc_400.filter3.BusMapper
Type: [PVBusMapper](#).

bp.utility_bus_map0
Type: [PVBusMapper](#).

bp.utility_bus_map1
Type: [PVBusMapper](#).

bp.utility_bus_map2
Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-R82 Cluster CT model.

Type: [Cluster_ARM_Cortex-R82](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core6

Type: PPUv1.

cluster0.DSU.PPU_core6.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core7

Type: PPUv1.

cluster0.DSU.PPU_core7.busslave

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu2.UTLB

Type: [TLB](#).

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu3.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4

ARM Cortex-R82 CT model.

Type: `ARM_Cortex-R82`.

cluster0.cpu4.UTLB

Type: `TLB`.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu4.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu4.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu5

ARM Cortex-R82 CT model.

Type: `ARM_Cortex-R82`.

cluster0.cpu5.UTLB

Type: `TLB`.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu5.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu5.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6**

ARM Cortex-R82 CT model.

Type: `ARM_Cortex-R82`.**cluster0.cpu6.UTLB**Type: `TLB`.**cluster0.cpu6.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu6.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu6.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7**

ARM Cortex-R82 CT model.

Type: `ARM_Cortex-R82`.**cluster0.cpu7.UTLB**Type: `TLB`.**cluster0.cpu7.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu7.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu7.l1dcache
PV Cache.
Type: PVCache.

cluster0.cpu7.l1dcache.upstream[0]
Type: PVBusSlave.

cluster0.cpu7.l1icache
PV Cache.
Type: PVCache.

cluster0.cpu7.l1icache.upstream[0]
Type: PVBusSlave.

cluster0.ext_bus
Type: PVBusLogger.

cluster0.ext_bus.mapper
Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_t1
Type: GICv3Distributor.

l1ram0
RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

l1ram0.bus_slave

Type: `PVBusSlave`.

pctl

Base Platforms Power Controller.

Type: `Base_PowerController`.

pctl.busslave

Type: `PVBusSlave`.

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

13.6 FVP_BaseR_Cortex-R82AE

List of instances in FVP_BaseR_Cortex-R82AE.

FVP_BaseR_Cortex-R82AE instances

address_map_terminator

Type: [PVBusMapper](#).

address_remap_switch

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

address_remap_switch.pvbus_mapper

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave
Type: [PVBusSlave](#).

bp.dram_limiter
Type: [PVBusMapper](#).

bp.dummy_local_dap_rom
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave
Type: [PVBusSlave](#).

bp.dummy_ram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_ram.bus_slave
Type: [PVBusSlave](#).

bp.dummy_usb
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.ns_dram.bus_slave`

Type: `PVBusSlave`.

`bp.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`bp.pl011_uart0.busslave`

Type: `PVBusSlave`.

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-R82AE Cluster CT model.

Type: [Cluster_ARM_Cortex-R82AE](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core2

Type: PPUv1.

cluster0.DSU.PPU_core2.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core3

Type: PPUv1.

cluster0.DSU.PPU_core3.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core4

Type: PPUv1.

cluster0.DSU.PPU_core4.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core5

Type: PPUv1.

cluster0.DSU.PPU_core5.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core6

Type: PPUv1.

cluster0.DSU.PPU_core6.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core7

Type: PPUv1.

cluster0.DSU.PPU_core7.busslave

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-R82AE CT model.

Type: [ARM_Cortex-R82AE](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-R82AE CT model.

Type: [ARM_Cortex-R82AE](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-R82AE CT model.

Type: [ARM_Cortex-R82AE](#).

cluster0.cpu2.UTLB

Type: [TLB](#).

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-R82AE CT model.

Type: [ARM_Cortex-R82AE](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu3.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4

ARM Cortex-R82AE CT model.

Type: `ARM_Cortex-R82AE`.

cluster0.cpu4.UTLB

Type: `TLB`.

cluster0.cpu4.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu4.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu4.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu4.l1icache

PV Cache.

Type: `PVCache`.

cluster0.cpu4.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster0.cpu5

ARM Cortex-R82AE CT model.

Type: `ARM_Cortex-R82AE`.

cluster0.cpu5.UTLB

Type: `TLB`.

cluster0.cpu5.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster0.cpu5.gicv3_cpu_if

Type: `GICv3CPUInterface`.

cluster0.cpu5.l1dcache

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu5.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu5.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6**

ARM Cortex-R82AE CT model.

Type: `ARM_Cortex-R82AE`.**cluster0.cpu6.UTLB**Type: `TLB`.**cluster0.cpu6.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster0.cpu6.gicv3_cpu_if**Type: `GICv3CPUInterface`.**cluster0.cpu6.l1dcache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu6.l1icache**

PV Cache.

Type: `PVCache`.**cluster0.cpu6.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster0.cpu7**

ARM Cortex-R82AE CT model.

Type: `ARM_Cortex-R82AE`.**cluster0.cpu7.UTLB**Type: `TLB`.**cluster0.cpu7.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster0.cpu7.gicv3_cpu_if
Type: GICv3CPUInterface.

cluster0.cpu7.l1dcache
PV Cache.
Type: PVCache.

cluster0.cpu7.l1dcache.upstream[0]
Type: PVBusSlave.

cluster0.cpu7.l1icache
PV Cache.
Type: PVCache.

cluster0.cpu7.l1icache.upstream[0]
Type: PVBusSlave.

cluster0.ext_bus
Type: PVBusLogger.

cluster0.ext_bus.mapper
Type: PVBusMapper.

cluster0.gic_cpuif_decoder_cluster
Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache
PV Cache.
Type: PVCache.

cluster0.l2_cache.upstream[0]
Type: PVBusSlave.

cluster0.l2_cache.upstream[10]
Type: PVBusSlave.

cluster0.l2_cache.upstream[11]
Type: PVBusSlave.

cluster0.l2_cache.upstream[12]
Type: PVBusSlave.

cluster0.l2_cache.upstream[13]
Type: PVBusSlave.

cluster0.l2_cache.upstream[14]
Type: PVBusSlave.

cluster0.l2_cache.upstream[15]
Type: PVBusSlave.

cluster0.l2_cache.upstream[16]
Type: PVBusSlave.

cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBUSLogger](#).

dapmemlogger.mapper

Type: [PVBUSMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBUSMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_0_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_1
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_2
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_3
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_4
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_4_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_5
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_5_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_6
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_6_0
Type: GICv3Redistributor.

gic_distributor.rd_0_0_7
Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_7_0
Type: GICv3Redistributor.

gic_distributor.rd_t1
Type: GICv3Distributor.

l1ram0
RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

l1ram0.bus_slave

Type: `PVBusSlave`.

pctl

Base Platforms Power Controller.

Type: `Base_PowerController`.

pctl.busslave

Type: `PVBusSlave`.

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

pctl.utility_bus0

Type: `PVBusMaster`.

pctl.utility_bus1

Type: `PVBusMaster`.

pctl.utility_bus2

Type: `PVBusMaster`.

pctl.utility_bus3

Type: `PVBusMaster`.

13.7 FVP_BaseR_Cortex-R82x1

List of instances in FVP_BaseR_Cortex-R82x1.

FVP_BaseR_Cortex-R82x1 instances

address_map_terminator

Type: [PVBusMapper](#).

address_remap_switch

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

address_remap_switch.pvbus_mapper

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBusSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase2

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase3

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase4

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase5

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase6

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase7

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave
Type: [PVBusSlave](#).

bp.dram_limiter
Type: [PVBusMapper](#).

bp.dummy_local_dap_rom
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave
Type: [PVBusSlave](#).

bp.dummy_ram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_ram.bus_slave
Type: [PVBusSlave](#).

bp.dummy_usb
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.ns_dram.bus_slave`

Type: `PVBusSlave`.

`bp.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`bp.pl011_uart0.busslave`

Type: `PVBusSlave`.

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.trusted_watchdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.trusted_watchdog.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.tzc_400

TrustZone Address Space Controller.

Type: [TzC_400](#).

bp.tzc_400.apbslave[0]

Type: [PVBusSlave](#).

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifierType: [PVBusMapper](#).**bp.virtio_rng**

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).**bp.virtio_rng.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice**

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).**bp.virtio_blockdevice.dma_master**Type: [PVBusMaster](#).**bp.virtio_blockdevice_labeller**Type: [Labeller](#).**bp.virtio_blockdevice_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.virtio_p9device**

virtio P9 server.

Type: [VirtioP9Device](#).**bp.virtio_p9device.mmio_slave**Type: [PVBusSlave](#).**bp.virtio_p9device.virtio_master**Type: [PVBusMaster](#).**bp.virtio_p9device_labeller**Type: [Labeller](#).**bp.virtio_p9device_labeller.pvbusmodifier**Type: [PVBusMapper](#).**bp.vis**

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).**bp.vis.recorder**

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).**bp.vis.recorder.playbackDivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-R82 Cluster CT model.

Type: [Cluster_ARM_Cortex-R82](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-R82 CT model.

Type: ARM_Cortex-R82.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: Pvcache.

cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster0.ext_bus

Type: PVBusLogger.

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: [GICv3CPUInterfaceDecoder](#).

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: [GICv3Redistributor](#).

gic_distributor.rd_t1

Type: [GICv3Distributor](#).

l1ram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

l1ram0.bus_slave

Type: [PVBusSlave](#).

pctl

Base Platforms Power Controller.

Type: [Base_PowerController](#).

pctl.busslave

Type: [PVBusSlave](#).

pctl.timer_reset

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

13.8 FVP_BaseR_Cortex-R82x2

List of instances in FVP_BaseR_Cortex-R82x2.

FVP_BaseR_Cortex-R82x2 instances

address_map_terminator

Type: [PVBusMapper](#).

address_remap_switch

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

address_remap_switch.pvbus_mapper

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: [BasePlatformPeripherals](#).

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_0_1.busslave

Type: [PVBusSlave](#).

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

bp.Timer_2_3.busslave

Type: [PVBUSSlave](#).

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase1

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase10

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase11

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase12

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase13

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase14

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBusSlave](#).

bp.ap_refclk.busbase9

Type: [PVBusSlave](#).

bp.ap_refclk.buscctlbase

Type: [PVBusSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave
Type: [PVBusSlave](#).

bp.dram_limiter
Type: [PVBusMapper](#).

bp.dummy_local_dap_rom
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave
Type: [PVBusSlave](#).

bp.dummy_ram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_ram.bus_slave
Type: [PVBusSlave](#).

bp.dummy_usb
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map

Type: [PVBusMapper](#).

bp.flash1.mbs

Type: [PVBusSlave](#).

bp.flash1.rmbs

Type: [PVBusSlave](#).

bp.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.generic_watchdog

ARM Generic Watchdog.

Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase

Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`bp.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`bp.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`bp.mpe`

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: `PVMemoryProtectionEngine`.

`bp.mpe.mapper`

Type: `PVBusMapper`.

`bp.nontrustedrom`

Intel Strata Flash J3 LISA+ model.

Type: `IntelStrataFlashJ3`.

`bp.nontrustedrom.map`

Type: `PVBusMapper`.

`bp.nontrustedrom.mbs`

Type: `PVBusSlave`.

`bp.nontrustedrom.rmbs`

Type: `PVBusSlave`.

`bp.nontrustedromloader`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`bp.ns_dram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`bp.ns_dram.bus_slave`

Type: `PVBusSlave`.

`bp.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`bp.pl011_uart0.busslave`

Type: `PVBusSlave`.

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

bp.pl041_aaci.busslave

Type: [PVBUSSlave](#).

bp.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PLO50_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave

Type: [PVBusSlave](#).

bp.refcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

bp.reset_or

Or Gate.

Type: [OrGate](#).

bp.rl_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rl_dram.bus_slave

Type: [PVBusSlave](#).

bp.rt_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.rt_dram.bus_slave

Type: [PVBusSlave](#).

bp.s_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.s_dram.bus_slave

Type: [PVBusSlave](#).

bp.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureDRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.secureSRAM.bus_slave

Type: [PVBusSlave](#).

bp.secureSRAM_limiter

Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmb

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

bp.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBUSSlave](#).

bp.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbusslave

Type: [PVBUSSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: `NonVolatileCounter`.

bp.trusted_nv_counter.pvbuslave

Type: `PVBusSlave`.

bp.trusted_rng

Random Number Generator unit.

Type: `RandomNumberGenerator`.

bp.trusted_rng.pvbuslave

Type: `PVBusSlave`.

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

bp.trusted_watchdog.busslave

Type: `PVBusSlave`.

bp.trusted_watchdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.trusted_watchdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.trusted_watchdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.trusted_watchdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.tzc_400

TrustZone Address Space Controller.

Type: `TZC_400`.

bp.tzc_400.apbslave[0]

Type: `PVBusSlave`.

bp.tzc_400.filter0
Type: `filter0`.

bp.tzc_400.filter0.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter1
Type: `filter1`.

bp.tzc_400.filter1.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter2
Type: `filter2`.

bp.tzc_400.filter2.BusMapper
Type: `PVBusMapper`.

bp.tzc_400.filter3
Type: `filter3`.

bp.tzc_400.filter3.BusMapper
Type: `PVBusMapper`.

bp.utility_bus_map0
Type: `PVBusMapper`.

bp.utility_bus_map1
Type: `PVBusMapper`.

bp.utility_bus_map2
Type: `PVBusMapper`.

bp.utility_bus_map3
Type: `PVBusMapper`.

bp.ve_sysregs
Type: `VE_SysRegs`.

bp.ve_sysregs.busslave
Type: `PVBusSlave`.

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: `VEDCC`.

bp.virtio_net
VirtioNet device over MMIO transport.
Type: `VirtioNetMMIO`.

bp.virtio_net.dma_master
Type: `PVBusMaster`.

bp.virtio_net_labeller
Type: `Labeller`.

bp.virtio_net_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_rng

VirtioEntropy device over MMIO transport.

Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice

VirtioBlock device over MMIO transport.

Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master

Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller

Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

bp.virtio_p9device_labeller

Type: [Labeller](#).

bp.virtio_p9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBUSSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCACHE](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-R82 Cluster CT model.

Type: [Cluster_ARM_Cortex-R82](#).

cluster0.AMU

Type: [PVBUSLogger](#).

cluster0.AMU.mapper

Type: [PVBUSMapper](#).

cluster0.DAP

Type: [PVBUSLogger](#).

cluster0.DAP.mapper

Type: [PVBUSMapper](#).

cluster0.DSU

Type: DSU.

cluster0.DSU.PPU_cluster

Type: PPUv1.

cluster0.DSU.PPU_cluster.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core0

Type: PPUv1.

cluster0.DSU.PPU_core0.busslave

Type: PVBusSlave.

cluster0.DSU.PPU_core1

Type: PPUv1.

cluster0.DSU.PPU_core1.busslave

Type: PVBusSlave.

cluster0.DSU.utility_slave[0]

Type: PVBusSlave.

cluster0.MMAP

Type: PVBusLogger.

cluster0.MMAP.mapper

Type: PVBusMapper.

cluster0.cpu0

ARM Cortex-R82 CT model.

Type: ARM_Cortex-R82.

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu0.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu0.l1dcache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster0.cpu0.l1icache

PV Cache.

Type: PVCache.

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster0.cpu1.gicv3_cpu_if

Type: GICv3CPUInterface.

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.ext_bus

Type: [PVBusLogger](#).

cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

cluster0.gic_cpuif_decoder_cluster

Type: GICv3CPUInterfaceDecoder.

cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[13]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[14]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[15]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[16]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[1]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[2]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.12_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmasterType: [PVBusMaster](#).**gic_distributor**

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).**gic_distributor.rd_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_0_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_0_0_1**Type: [GICv3RedistributorInternal](#).**gic_distributor.rd_0_0_1_0**Type: [GICv3Redistributor](#).**gic_distributor.rd_tl**Type: [GICv3Distributor](#).**l1ram0**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**l1ram0.bus_slave**Type: [PVBusSlave](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

pctl.timer_reset.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

13.9 FVP_BaseR_Cortex-R82x4

List of instances in FVP_BaseR_Cortex-R82x4.

FVP_BaseR_Cortex-R82x4 instances

address_map_terminator

Type: [PVBusMapper](#).

address_remap_switch

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

address_remap_switch.pvbus_mapper

Type: [PVBusMapper](#).

address_swapper

Type: [PVBusMapper](#).

bp

Peripherals and address map for the Base Platform.

Type: `BasePlatformPeripherals`.

bp.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_0_1.busslave

Type: `PVBusSlave`.

bp.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

bp.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

bp.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

bp.Timer_2_3.busslave

Type: `PVBusSlave`.

bp.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

bp.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

bp.ap_refclk

ARM Generic Timer.

Type: [MemoryMappedGenericTimer](#).

bp.ap_refclk.busbase0

Type: [PVBusSlave](#).

bp.ap_refclk.busbase1

Type: [PVBusSlave](#).

bp.ap_refclk.busbase10

Type: [PVBusSlave](#).

bp.ap_refclk.busbase11

Type: [PVBusSlave](#).

bp.ap_refclk.busbase12

Type: [PVBusSlave](#).

bp.ap_refclk.busbase13

Type: [PVBusSlave](#).

bp.ap_refclk.busbase14

Type: [PVBusSlave](#).

bp.ap_refclk.busbase15

Type: [PVBusSlave](#).

bp.ap_refclk.busbase2

Type: [PVBusSlave](#).

bp.ap_refclk.busbase3

Type: [PVBusSlave](#).

bp.ap_refclk.busbase4

Type: [PVBusSlave](#).

bp.ap_refclk.busbase5

Type: [PVBusSlave](#).

bp.ap_refclk.busbase6

Type: [PVBusSlave](#).

bp.ap_refclk.busbase7

Type: [PVBusSlave](#).

bp.ap_refclk.busbase8

Type: [PVBUSSlave](#).

bp.ap_refclk.busbase9

Type: [PVBUSSlave](#).

bp.ap_refclk.busctlbase

Type: [PVBUSSlave](#).

bp.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

bp.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock300MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock32KHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.coresight_modifier

Type: [PVBusMapper](#).

bp.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc.bus_slave

Type: [PVBusSlave](#).

bp.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dmc_phy.bus_slave

Type: [PVBusSlave](#).

bp.dram_limiter

Type: [PVBusMapper](#).

bp.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

bp.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_ram.bus_slave

Type: [PVBusSlave](#).

bp.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.dummy_usb.bus_slave
Type: [PVBusSlave](#).

bp.exclusive_monitor
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

bp.exclusive_monitor.bus_mapper
Type: [PVBusMapper](#).

bp.fixed_security_map
Type: [PVBusMapper](#).

bp.flash0
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash0.map
Type: [PVBusMapper](#).

bp.flash0.mbs
Type: [PVBusSlave](#).

bp.flash0.rmbs
Type: [PVBusSlave](#).

bp.flash1
Intel Strata Flash J3 LISA+ model.
Type: [IntelStrataFlashJ3](#).

bp.flash1.map
Type: [PVBusMapper](#).

bp.flash1.mbs
Type: [PVBusSlave](#).

bp.flash1.rmbs
Type: [PVBusSlave](#).

bp.flashloader0
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.flashloader1
A device that can preload a gzipped image into flash at startup.
Type: [FlashLoader](#).

bp.generic_watchdog
ARM Generic Watchdog.
Type: [MemoryMappedGenericWatchdog](#).

bp.generic_watchdog.busctlbase
Type: [PVBusSlave](#).

bp.generic_watchdog.busrefbase

Type: [PVBusSlave](#).

bp.hdlcd0

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

bp.hdlcd0.busmaster

Type: [PVBusMaster](#).

bp.hdlcd0.busslave

Type: [PVBusSlave](#).

bp.hdlcd0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.hdlcd0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.hdlcd0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.hdlcd0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.hdlcd0_labeller

Type: [Labeller](#).

bp.hdlcd0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

bp.lcd_security_map

Type: [PVBusMapper](#).

bp.ls64_testing_fifo

FIFO peripheral supporting LS64 accesses for testing purposes.

Type: [LS64TestingFIFO](#).

bp.ls64_testing_fifo.pvbusslave

Type: [PVBusSlave](#).

bp.mmc

Generic Multimedia Card.

Type: [MMC](#).

bp.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.mpe

Encrypt memory transactions for each encryption context with an independent key to prevent mismatch access.

Type: [PVMemoryProtectionEngine](#).

bp.mpe.mapper

Type: [PVBusMapper](#).

bp.nontrustedrom

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.nontrustedrom.map

Type: [PVBusMapper](#).

bp.nontrustedrom.mbs

Type: [PVBusSlave](#).

bp.nontrustedrom.rmbs

Type: [PVBusSlave](#).

bp.nontrustedromloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.ns_dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.ns_dram.bus_slave

Type: [PVBusSlave](#).

bp.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart0.busslave

Type: [PVBusSlave](#).

bp.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

bp.pl011_uart1.busslave

Type: [PVBusSlave](#).

bp.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart2

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart2.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl011_uart3

ARM PrimeCell UART(PLO11).

Type: [PL011_Uart](#).

bp.pl011_uart3.busslave

Type: [PVBUSSlave](#).

bp.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

bp.pl031_rtc.busslave

Type: [PVBUSSlave](#).

bp.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

bp.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: `PL041_AACI`.

bp.pl041_aaci.busslave

Type: `PVBusSlave`.

bp.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

bp.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

bp.pl041_aaci.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

bp.pl041_aaci.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

bp.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

bp.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

bp.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

bp.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

bp.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

bp.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.pl111_clcd_labeller

Type: [Labeller](#).

bp.pl111_clcd_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

bp.pl180_mci.busslave

Type: [PVBusSlave](#).

bp.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

bp.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

bp.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.psram.bus_slave
Type: [PVBusSlave](#).

bp.refcounter
Memory Mapped Counter Module for Generic Timers.
Type: [MemoryMappedCounterModule](#).

bp.refcounter.pvbus_control_s[0]
Type: [PVBusSlave](#).

bp.refcounter.pvbus_read_s[0]
Type: [PVBusSlave](#).

bp.reset_or
Or Gate.
Type: [OrGate](#).

bp.rl_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rl_dram.bus_slave
Type: [PVBusSlave](#).

bp.rt_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.rt_dram.bus_slave
Type: [PVBusSlave](#).

bp.s_dram
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.s_dram.bus_slave
Type: [PVBusSlave](#).

bp.secureDRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureDRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

bp.secureSRAM.bus_slave
Type: [PVBusSlave](#).

bp.secureSRAM_limiter
Type: [PVBusMapper](#).

bp.secureflash

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

bp.secureflash.map

Type: [PVBusMapper](#).

bp.secureflash.mbs

Type: [PVBusSlave](#).

bp.secureflash.rmbs

Type: [PVBusSlave](#).

bp.secureflashloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

bp.sm5c_91c111

SM5C 91C111 ethernet controller.

Type: [SM5C_91C111](#).

bp.sm5c_91c111.SM5C_slave

Type: [PVBusSlave](#).

bp.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.sp805_wdog.busslave

Type: [PVBusSlave](#).

bp.sp805_wdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.sp805_wdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

bp.sp805_wdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

bp.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

bp.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.sram.bus_slave

Type: [PVBUSSlave](#).

bp.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

bp.trusted_key_storage

Trusted Root-Key Storage unit.

Type: [RootKeyStorage](#).

bp.trusted_key_storage.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_nv_counter

Trusted Non-Volatile Counter unit.

Type: [NonVolatileCounter](#).

bp.trusted_nv_counter.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_rng

Random Number Generator unit.

Type: [RandomNumberGenerator](#).

bp.trusted_rng.pvbuslave

Type: [PVBusSlave](#).

bp.trusted_watchdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

bp.trusted_watchdog.busslave

Type: [PVBusSlave](#).

bp.trusted_watchdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

bp.trusted_watchdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`bp.trusted_watchdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`bp.trusted_watchdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`bp.tzc_400`

TrustZone Address Space Controller.

Type: [TZC_400](#).

`bp.tzc_400.apbslave[0]`

Type: [PVBusSlave](#).

`bp.tzc_400.filter0`

Type: `filter0`.

`bp.tzc_400.filter0.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter1`

Type: `filter1`.

`bp.tzc_400.filter1.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter2`

Type: `filter2`.

`bp.tzc_400.filter2.BusMapper`

Type: [PVBusMapper](#).

`bp.tzc_400.filter3`

Type: `filter3`.

`bp.tzc_400.filter3.BusMapper`

Type: [PVBusMapper](#).

`bp.utility_bus_map0`

Type: [PVBusMapper](#).

`bp.utility_bus_map1`

Type: [PVBusMapper](#).

`bp.utility_bus_map2`

Type: [PVBusMapper](#).

bp.utility_bus_map3
Type: [PVBusMapper](#).

bp.ve_sysregs
Type: [vE_SysRegs](#).

bp.ve_sysregs.busslave
Type: [PVBusSlave](#).

bp.vedcc
Daughterboard Configuration Control (DCC).
Type: [vEDCC](#).

bp.virtio_net
VirtioNet device over MMIO transport.
Type: [VirtioNetMMIO](#).

bp.virtio_net.dma_master
Type: [PVBusMaster](#).

bp.virtio_net_labeller
Type: [Labeller](#).

bp.virtio_net_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_rng
VirtioEntropy device over MMIO transport.
Type: [VirtioEntropyMMIO](#).

bp.virtio_rng.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice
VirtioBlock device over MMIO transport.
Type: [virtioBlockMMIO](#).

bp.virtio_blockdevice.dma_master
Type: [PVBusMaster](#).

bp.virtio_blockdevice_labeller
Type: [Labeller](#).

bp.virtio_blockdevice_labeller.pvbusmodifier
Type: [PVBusMapper](#).

bp.virtio_p9device
virtio P9 server.
Type: [VirtioP9Device](#).

bp.virtio_p9device.mmio_slave
Type: [PVBusSlave](#).

bp.virtio_p9device.virtio_master
Type: [PVBusMaster](#).

bp.virtiop9device_labeller

Type: [Labeller](#).

bp.virtiop9device_labeller.pvbusmodifier

Type: [PVBusMapper](#).

bp.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

bp.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

bp.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

bp.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

bp.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

bp.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

bp.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

bp.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

bp.vram.bus_slave

Type: [PVBusSlave](#).

cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

cci400.cciinterconnect

Type: [PVCache](#).

cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

clockdivider0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cluster0

ARM Cortex-R82 Cluster CT model.

Type: [cluster_ARM_Cortex-R82](#).

cluster0.AMU

Type: [PVBusLogger](#).

cluster0.AMU.mapper

Type: [PVBusMapper](#).

cluster0.DAP

Type: [PVBusLogger](#).

cluster0.DAP.mapper

Type: [PVBusMapper](#).

cluster0.DSU

Type: [DSU](#).

cluster0.DSU.PPU_cluster

Type: [PPUv1](#).

cluster0.DSU.PPU_cluster.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core0

Type: [PPUv1](#).

cluster0.DSU.PPU_core0.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core1

Type: [PPUv1](#).

cluster0.DSU.PPU_core1.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core2

Type: [PPUv1](#).

cluster0.DSU.PPU_core2.busslave

Type: [PVBusSlave](#).

cluster0.DSU.PPU_core3

Type: [PPUv1](#).

cluster0.DSU.PPU_core3.busslave

Type: [PVBusSlave](#).

cluster0.DSU.utility_slave[0]

Type: [PVBusSlave](#).

cluster0.MMAP

Type: [PVBusLogger](#).

cluster0.MMAP.mapper

Type: [PVBusMapper](#).

cluster0.cpu0

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu0.UTLB

Type: TLB.

cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu0.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu1.UTLB

Type: TLB.

cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster0.cpu1.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu2.UTLB

Type: [TLB](#).

cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu2.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu2.l1dcache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

cluster0.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster0.cpu3

ARM Cortex-R82 CT model.

Type: [ARM_Cortex-R82](#).

cluster0.cpu3.UTLB

Type: [TLB](#).

cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster0.cpu3.gicv3_cpu_if

Type: [GICv3CPUInterface](#).

cluster0.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

cluster0.cpu3.l1dcache.upstream[0]
Type: `PVBusSlave`.

cluster0.cpu3.l1icache
PV Cache.
Type: `PVCache`.

cluster0.cpu3.l1icache.upstream[0]
Type: `PVBusSlave`.

cluster0.ext_bus
Type: `PVBusLogger`.

cluster0.ext_bus.mapper
Type: `PVBusMapper`.

cluster0.gic_cpuif_decoder_cluster
Type: `GICv3CPUInterfaceDecoder`.

cluster0.l2_cache
PV Cache.
Type: `PVCache`.

cluster0.l2_cache.upstream[0]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[10]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[11]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[12]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[13]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[14]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[15]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[16]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[1]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[2]
Type: `PVBusSlave`.

cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster0.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster0_labeller

Type: [Labeller](#).

cluster0_labeller.pvbusmodifier

Type: [PVBusMapper](#).

dapmemlogger

Bus Logger.

Type: [PVBusLogger](#).

dapmemlogger.mapper

Type: [PVBusMapper](#).

elfloader

ELF loader component.

Type: [ElfLoader](#).

elfloader.pvbus_busmaster

Type: [PVBusMaster](#).

gic_distributor

GIC Interrupt Redistribution Infrastructure component.

Type: [GIC_IRI](#).

gic_distributor.rd_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0

Type: [GICv3RedistributorInternal](#).

gic_distributor.rd_0_0_0_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_1

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_1_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_2

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_2_0

Type: GICv3Redistributor.

gic_distributor.rd_0_0_3

Type: GICv3RedistributorInternal.

gic_distributor.rd_0_0_3_0

Type: GICv3Redistributor.

gic_distributor.rd_tl

Type: GICv3Distributor.

llram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**llram0.bus_slave**Type: [PVBusSlave](#).**pctl**

Base Platforms Power Controller.

Type: [Base_PowerController](#).**pctl.busslave**Type: [PVBusSlave](#).**pctl.timer_reset**

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).**pctl.timer_reset.timer**

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

pctl.timer_reset.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

pctl.timer_reset.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

pctl.utility_bus0

Type: [PVBusMaster](#).

pctl.utility_bus1

Type: [PVBusMaster](#).

pctl.utility_bus2

Type: [PVBusMaster](#).

pctl.utility_bus3

Type: [PVBusMaster](#).

14. VE Platform FVPs

This chapter describes the VE Platform FVPs contained in the FVP Standard Library.

For the VE memory maps, see [VE memory map for Cortex-A series](#) and [VE memory map for Cortex-R series](#) in the Fast Models Reference Guide.

14.1 FVP_VE_Cortex-A15x1

List of instances in FVP_VE_Cortex-A15x1.

FVP_VE_Cortex-A15x1 instances

cluster

ARM Cortex-A15 Cluster CT model.

Type: `Cluster_ARM_Cortex-A15`.

cluster.Cortex-A15_GIC

Type: `GICv2`.

cluster.acp_mapper

Type: `PVBusMapper`.

cluster.cpu0

ARM Cortex-A15 CT model.

Type: `ARM_Cortex-A15`.

cluster.cpu0.DTLB

Type: `TLB`.

cluster.cpu0.ITLB

Type: `TLB`.

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.l1icache

PV Cache.

Type: `PVCache`.

`cluster.cpu0.l1icache.upstream[0]`

Type: `PVBusSlave`.

`cluster.ext_bus`

Type: `PVBusLogger`.

`cluster.ext_bus.mapper`

Type: `PVBusMapper`.

`cluster.l2_cache`

PV Cache.

Type: `PVCache`.

`cluster.l2_cache.upstream[0]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[10]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[11]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[12]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[13]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[14]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[15]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[16]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[1]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[2]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[3]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[4]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[5]`

Type: `PVBusSlave`.

`cluster.l2_cache.upstream[6]`

Type: `PVBusSlave`.

cluster.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBUSMapper](#).

motherboard.flash0.mbs

Type: [PVBUSSlave](#).

motherboard.flash0.rmbs

Type: [PVBUSSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBUSMapper](#).

motherboard.flash1.mbs

Type: [PVBUSSlave](#).

motherboard.flash1.rmbs

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

motherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

motherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

motherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtiop9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtiop9device.mmio_slave`

Type: [PVBusSlave](#).

motherboard.virtio9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.2 FVP_VE_Cortex-A15x1-A7x1

List of instances in FVP_VE_Cortex-A15x1-A7x1.

FVP_VE_Cortex-A15x1-A7x1 instances

coretile

Dual cluster ARM Cortex-A15x1 and ARM Cortex-A7x1 Core Tile.

Type: [ARM_Cortex_A15x1_A7x1_CT](#).

coretile.cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

coretile.cci400.cciinterconnect

Type: [PVCache](#).

coretile.cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

coretile.cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

coretile.cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

coretile.cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

coretile.cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

coretile.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.cluster0

ARM Cortex-A15 Cluster CT model.

Type: [Cluster_ARM_Cortex-A15](#).

coretile.cluster0.acp_mapper

Type: [PVBusMapper](#).

coretile.cluster0.cpu0

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

coretile.cluster0.cpu0.DTLB

Type: [TLB](#).

coretile.cluster0.cpu0.ITLB

Type: [TLB](#).

coretile.cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster0.cpu0.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

coretile.cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster0.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.ext_bus

Type: [PVBusLogger](#).

coretile.cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

coretile.cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

coretile.cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[13]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[14]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[15]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[5]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[6]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[7]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[8]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

coretile.cluster0.l2_flusher
Type: AsyncCacheFlushUnit.

coretile.cluster1
ARM Cortex-A7 Cluster CT model.
Type: Cluster_ARM_Cortex-A7.

coretile.cluster1.acp_mapper
Type: [PVBusMapper](#).

coretile.cluster1.cpu0
ARM Cortex-A7 CT model.
Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu0.DTLB
Type: TLB.

coretile.cluster1.cpu0.ITLB
Type: TLB.

coretile.cluster1.cpu0.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

coretile.cluster1.cpu0.itlb
TLB - instruction, data or unified.
Type: TlbCadi.

coretile.cluster1.cpu0.l1dcache
PV Cache.
Type: [PVCache](#).

coretile.cluster1.cpu0.l1dcache.upstream[0]
Type: [PVBusSlave](#).

coretile.cluster1.cpu0.l1icache
PV Cache.
Type: [PVCache](#).

coretile.cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.ext_bus

Type: [PVBusLogger](#).

coretile.cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

coretile.cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[10]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[11]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[12]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[13]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[14]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[15]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[16]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[6]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

coretile.dualclustersystemconfigurationblock

Dual Cluster System Configuration Block.

Type: [DualClusterSystemConfigurationBlock](#).

coretile.dualclustersystemconfigurationblock.pvbusslave

Type: [PVBUSSlave](#).

coretile.globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

coretile.globalcounter.pvbus_control_s[0]

Type: [PVBUSSlave](#).

coretile.globalcounter.pvbus_read_s[0]

Type: [PVBUSSlave](#).

coretile.v7_vgic

System VGIC architecture version v7.

Type: [v7_VGIC](#).

coretile.v7_vgic.vgic_bus_slave

Type: [PVBUSSlave](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCDC`.

daughterboard.hdlcd.busmaster

Type: `PVBusMaster`.

daughterboard.hdlcd.busslave

Type: `PVBusSlave`.

daughterboard.hdlcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

daughterboard.hdlcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

daughterboard.hdlcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

daughterboard.hdlcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: `VEInterruptMapper`.

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: `TZSwitch`.

daughterboard.nonsecure_region.pvbus_mapper

Type: `PVBusMapper`.

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

daughterboard.secureDRAM.bus_slave

Type: `PVBusSlave`.

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

`motherboard.clock100Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBusSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBusMapper](#).

`motherboard.flash0.mbs`

Type: [PVBusSlave](#).

`motherboard.flash0.rmbs`

Type: [PVBusSlave](#).

`motherboard.flash1`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash1.map`

Type: [PVBusMapper](#).

`motherboard.flash1.mbs`

Type: [PVBusSlave](#).

`motherboard.flash1.rmbs`

Type: [PVBusSlave](#).

`motherboard.flashloader0`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.flashloader1`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.hostbridge`

Host Socket Interface Component.

Type: [HostBridge](#).

`motherboard.mmc`

Generic Multimedia Card.

Type: [MMC](#).

`motherboard.mmc.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.mmc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBusSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl031_rtc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl031_rtc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi1.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl111_clcd`

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd`

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd.busmaster`

Type: [PVBusMaster](#).

`motherboard.pl111_clcd.pl11x_clcd.busslave`

Type: [PVBusSlave](#).

`motherboard.pl111_clcd.pl11x_clcd.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBUSSlave](#).

motherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

motherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBUSSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [vE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.3 FVP_VE_Cortex-A15x2

List of instances in FVP_VE_Cortex-A15x2.

FVP_VE_Cortex-A15x2 instances

cluster

ARM Cortex-A15 Cluster CT model.

Type: [Cluster_ARM_Cortex-A15](#).

cluster.Cortex-A15_GIC

Type: [GICv2](#).

cluster.acp_mapper

Type: [PVBusMapper](#).

cluster.cpu0

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

cluster.cpu0.DTLB

Type: [TLB](#).

cluster.cpu0.ITLB

Type: [TLB](#).

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu1

ARM Cortex-A15 CT model.

Type: `ARM_Cortex-A15`.

cluster.cpu1.DTLB

Type: `TLB`.

cluster.cpu1.ITLB

Type: `TLB`.

cluster.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu1.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu1.l1icache

PV Cache.

Type: `PVCache`.

cluster.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster.ext_bus

Type: `PVBusLogger`.

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.l2_cache

PV Cache.

Type: [PVCache](#).

cluster.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapperType: [PVBusMapper](#).**daughterboard.dram_limit_4**

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).**daughterboard.dram_limit_4.pvbus_mapper**Type: [PVBusMapper](#).**daughterboard.dram_limit_8**

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).**daughterboard.dram_limit_8.pvbus_mapper**Type: [PVBusMapper](#).**daughterboard.exclusive_monitor**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).**daughterboard.exclusive_monitor.bus_mapper**Type: [PVBusMapper](#).**daughterboard.hdlcd**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).**daughterboard.hdlcd.busmaster**Type: [PVBusMaster](#).**daughterboard.hdlcd.busslave**Type: [PVBusSlave](#).**daughterboard.hdlcd.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**daughterboard.hdlcd.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBusSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBusMapper](#).

`motherboard.flash0.mbs`

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBUSSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBUSMapper](#).

motherboard.flash1.mbs

Type: [PVBUSSlave](#).

motherboard.flash1.rmbs

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.pl011_uart2.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl011_uart2.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl011_uart2.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`motherboard.pl011_uart3.busslave`

Type: `PVBusSlave`.

`motherboard.pl011_uart3.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.pl011_uart3.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBusSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

motherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.sp805_wdog.clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.sp805_wdog.clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

`motherboard.sp810_sysctrl.busslave`

Type: [PVBUSSlave](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtio_blockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtio_blockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtio_blockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtio_p9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtio_p9device.mmio_slave`

Type: [PVBusSlave](#).

`motherboard.virtio_p9device.virtio_master`

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: `VEVisualisation`.

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: `VisEventRecorder`.

motherboard.vis.recorder.playbackDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

motherboard.vis.recorder.playbackTimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

motherboard.vis.recorder.playbackTimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

motherboard.vis.recorder.playbackTimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

motherboard.vis.recorder.playbackTimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

motherboard.vis.recorder.recordingDivider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.4 FVP_VE_Cortex-A15x2-A7x2

List of instances in FVP_VE_Cortex-A15x2-A7x2.

FVP_VE_Cortex-A15x2-A7x2 instances

coretile

Dual cluster ARM Cortex-A15x2 and ARM Cortex-A7x2 Core Tile.

Type: [ARM_Cortex_A15x2_A7x2_CT](#).

coretile.cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

coretile.cci400.cciinterconnect

Type: [PVCache](#).

coretile.cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

coretile.cci400.cciregisters.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

coretile.cci400.cciregisters.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

coretile.cci400.cciregisters.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

coretile.cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

coretile.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.cluster0

ARM Cortex-A15 Cluster CT model.

Type: [Cluster_ARM_Cortex-A15](#).

coretile.cluster0.acp_mapper

Type: [PVBusMapper](#).

coretile.cluster0.cpu0

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

coretile.cluster0.cpu0.DTLB

Type: [TLB](#).

coretile.cluster0.cpu0.ITLB

Type: [TLB](#).

coretile.cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster0.cpu0.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster0.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

coretile.cluster0.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.

coretile.cluster0.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.cpu1

ARM Cortex-A15 CT model.

Type: `ARM_Cortex-A15`.

coretile.cluster0.cpu1.DTLB

Type: `TLB`.

coretile.cluster0.cpu1.ITLB

Type: `TLB`.

coretile.cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

coretile.cluster0.cpu1.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

coretile.cluster0.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

coretile.cluster0.cpu1.l1dcache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.cpu1.l1icache

PV Cache.

Type: `PVCache`.

coretile.cluster0.cpu1.l1icache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.ext_bus

Type: `PVBusLogger`.

coretile.cluster0.ext_bus.mapper

Type: `PVBusMapper`.

coretile.cluster0.l2_cache

PV Cache.

Type: `PVCache`.

coretile.cluster0.l2_cache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[10]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[11]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[12]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[13]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[14]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[15]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

coretile.cluster0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

coretile.cluster1

ARM Cortex-A7 Cluster CT model.

Type: [Cluster_ARM_Cortex-A7](#).

coretile.cluster1.acp_mapper

Type: [PVBUSMapper](#).

coretile.cluster1.cpu0

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu0.DTLB

Type: TLB.

coretile.cluster1.cpu0.ITLB

Type: TLB.

coretile.cluster1.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu0.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu0.l1dcache

PV Cache.

Type: PVCache.

coretile.cluster1.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

coretile.cluster1.cpu0.l1icache

PV Cache.

Type: PVCache.

coretile.cluster1.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

coretile.cluster1.cpu1

ARM Cortex-A7 CT model.

Type: ARM_Cortex-A7.

coretile.cluster1.cpu1.DTLB

Type: TLB.

coretile.cluster1.cpu1.ITLB

Type: TLB.

coretile.cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu1.l1dcache

PV Cache.

Type: PVCache.

coretile.cluster1.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

coretile.cluster1.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.ext_bus

Type: [PVBusLogger](#).

coretile.cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

coretile.cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[10]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[11]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[12]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[13]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[14]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[15]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[16]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[5]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

coretile.dualclustersystemconfigurationblock

Dual Cluster System Configuration Block.

Type: [DualClusterSystemConfigurationBlock](#).

coretile.dualclustersystemconfigurationblock.pvbusslave

Type: [PVBUSSlave](#).

coretile.globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

coretile.globalcounter.pvbus_control_s[0]

Type: [PVBUSSlave](#).

coretile.globalcounter.pvbus_read_s[0]

Type: [PVBUSSlave](#).

coretile.v7_vgic

System VGIC architecture version v7.

Type: [v7_VGIC](#).

coretile.v7_vgic.vgic_bus_slave

Type: [PVBUSSlave](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow [TrustZone](#) secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl011_uart0.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl011_uart0.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`motherboard.pl011_uart1.busslave`

Type: `PVBusSlave`.

`motherboard.pl011_uart1.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.pl011_uart1.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.pl011_uart1.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart2.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart2.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBusSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBusSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBUSSlave](#).

motherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

motherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

`motherboard.sp810_sysctrl.busslave`

Type: `PVBusSlave`.

`motherboard.sp810_sysctrl.clkdiv_clk0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.terminal_0`

Telnet terminal interface.

Type: `TelnetTerminal`.

`motherboard.terminal_1`

Telnet terminal interface.

Type: `TelnetTerminal`.

`motherboard.terminal_2`

Telnet terminal interface.

Type: `TelnetTerminal`.

`motherboard.terminal_3`

Telnet terminal interface.

Type: `TelnetTerminal`.

`motherboard.ve_sysregs`

Type: `VE_SysRegs`.

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtio_blockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtio_blockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtio_blockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtio_p9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtio_p9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.vis.recorder.playbackTimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.vis.recorder.recordingDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.vram.bus_slave`

Type: [PVBusSlave](#).

14.5 FVP_VE_Cortex-A15x4

List of instances in FVP_VE_Cortex-A15x4.

FVP_VE_Cortex-A15x4 instances

`cluster`

ARM Cortex-A15 Cluster CT model.

Type: [cluster_ARM_Cortex-A15](#).

`cluster.Cortex-A15_GIC`

Type: [GICv2](#).

`cluster.acp_mapper`

Type: [PVBusMapper](#).

`cluster.cpu0`

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

`cluster.cpu0.DTLB`

Type: [TLB](#).

cluster.cpu0.ITLB

Type: TLB.

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu0.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu0.l1icache

PV Cache.

Type: pVCache.

cluster.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster.cpu1

ARM Cortex-A15 CT model.

Type: ARM_Cortex-A15.

cluster.cpu1.DTLB

Type: TLB.

cluster.cpu1.ITLB

Type: TLB.

cluster.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu1.l1icache

PV Cache.

Type: pVCache.

cluster.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu2

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

cluster.cpu2.DTLB

Type: TLB.

cluster.cpu2.ITLB

Type: TLB.

cluster.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu2.l1dcache

PV Cache.

Type: [pVCache](#).

cluster.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu2.l1icache

PV Cache.

Type: [pVCache](#).

cluster.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu3

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

cluster.cpu3.DTLB

Type: TLB.

cluster.cpu3.ITLB

Type: TLB.

cluster.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu3.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu3.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu3.l1icache

PV Cache.

Type: `PVCache`.

cluster.cpu3.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster.ext_bus

Type: `PVBusLogger`.

cluster.ext_bus.mapper

Type: `PVBusMapper`.

cluster.l2_cache

PV Cache.

Type: `PVCache`.

cluster.l2_cache.upstream[0]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[10]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[11]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[12]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[13]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[14]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[15]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[16]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[1]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[2]

Type: `PVBusSlave`.

cluster.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

daughterboard.dmc_phy.bus_slave

Type: `PVBusSlave`.

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

daughterboard.dram.bus_slave

Type: `PVBusSlave`.

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: `TZSwitch`.

daughterboard.dram_aliased.pvbus_mapper

Type: `PVBusMapper`.

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: `TZSwitch`.

daughterboard.dram_limit_4.pvbus_mapper

Type: `PVBusMapper`.

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: `TZSwitch`.

daughterboard.dram_limit_8.pvbus_mapper

Type: `PVBusMapper`.

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

daughterboard.exclusive_monitor.bus_mapper

Type: `PVBusMapper`.

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: `PL370_HDLCD`.

daughterboard.hdlcd.busmaster

Type: `PVBusMaster`.

daughterboard.hdlcd.busslave

Type: `PVBusSlave`.

daughterboard.hdlcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.counter0`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.Timer_0_1.counter1`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.Timer_2_3`

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

`motherboard.Timer_2_3.busslave`

Type: [PVBusSlave](#).

`motherboard.Timer_2_3.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_local_dap_rom.bus_slave

Type: [PVBUSSlave](#).

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBUSSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBUSlave](#).

`motherboard.pl011_uart1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [`SchedulerThread`](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [`SchedulerThreadEvent`](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [`PL180_MCI`](#).

`motherboard.pl180_mci.busslave`

Type: [`PVBusSlave`](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: [`PS2Keyboard`](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread`](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.psram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.smisc_91c111`

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

`motherboard.smc_91c111.SMSC_slave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog`

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

`motherboard.sp805_wdog.busslave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.sp805_wdog.clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

`motherboard.sp810_sysctrl.busslave`

Type: `PVBusSlave`.

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtiop9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtiop9device.mmio_slave`

Type: [PVBusSlave](#).

`motherboard.virtiop9device.virtio_master`

Type: [PVBusMaster](#).

`motherboard.vis`

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

`motherboard.vis.recorder`

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

`motherboard.vis.recorder.playbackDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vis.recorder.playbackTimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.vis.recorder.playbackTimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBUSSlave](#).

14.6 FVP_VE_Cortex-A15x4-A7x4

List of instances in FVP_VE_Cortex-A15x4-A7x4.

FVP_VE_Cortex-A15x4-A7x4 instances

coretile

Dual cluster ARM Cortex-A15x4 and ARM Cortex-A7x4 Core Tile.

Type: [ARM_Cortex_A15x4_A7x4_CT](#).

coretile.cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

coretile.cci400.cciinterconnect

Type: [PVCACHE](#).

coretile.cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

coretile.cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

coretile.cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

coretile.cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

coretile.cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

coretile.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.cluster0

ARM Cortex-A15 Cluster CT model.

Type: [Cluster_ARM_Cortex-A15](#).

coretile.cluster0.acp_mapper

Type: [PVBusMapper](#).

coretile.cluster0.cpu0

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

coretile.cluster0.cpu0.DTLB

Type: TLB.

coretile.cluster0.cpu0.ITLB

Type: TLB.

coretile.cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster0.cpu0.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).**coretile.cluster0.cpu0.l1dcache**

PV Cache.

Type: [PvCache](#).**coretile.cluster0.cpu0.l1dcache.upstream[0]**Type: [PvBusSlave](#).**coretile.cluster0.cpu0.l1icache**

PV Cache.

Type: [PvCache](#).**coretile.cluster0.cpu0.l1icache.upstream[0]**Type: [PvBusSlave](#).**coretile.cluster0.cpu1**

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).**coretile.cluster0.cpu1.DTLB**Type: [TLB](#).**coretile.cluster0.cpu1.ITLB**Type: [TLB](#).**coretile.cluster0.cpu1.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**coretile.cluster0.cpu1.itlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**coretile.cluster0.cpu1.l1dcache**

PV Cache.

Type: [PvCache](#).**coretile.cluster0.cpu1.l1dcache.upstream[0]**Type: [PvBusSlave](#).**coretile.cluster0.cpu1.l1icache**

PV Cache.

Type: [PvCache](#).**coretile.cluster0.cpu1.l1icache.upstream[0]**Type: [PvBusSlave](#).**coretile.cluster0.cpu2**

ARM Cortex-A15 CT model.

Type: [ARM_Cortex-A15](#).

coretile.cluster0.cpu2.DTLB

Type: TLB.

coretile.cluster0.cpu2.ITLB

Type: TLB.

coretile.cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu2.l1dcache

PV Cache.

Type: PVCache.

coretile.cluster0.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu2.l1icache

PV Cache.

Type: PVCache.

coretile.cluster0.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu3

ARM Cortex-A15 CT model.

Type: ARM_Cortex-A15.

coretile.cluster0.cpu3.DTLB

Type: TLB.

coretile.cluster0.cpu3.ITLB

Type: TLB.

coretile.cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu3.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu3.l1dcache

PV Cache.

Type: PVCache.

coretile.cluster0.cpu3.l1dcache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu3.l1icache

PV Cache.

Type: `PVCache`.

coretile.cluster0.cpu3.l1icache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.ext_bus

Type: `PVBusLogger`.

coretile.cluster0.ext_bus.mapper

Type: `PVBusMapper`.

coretile.cluster0.l2_cache

PV Cache.

Type: `PVCache`.

coretile.cluster0.l2_cache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[10]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[11]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[12]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[13]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[14]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[15]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[16]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[1]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[2]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[3]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[4]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[5]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[6]
Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[7]
Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[8]
Type: [PVBUSSlave](#).

coretile.cluster0.l2_cache.upstream[9]
Type: [PVBUSSlave](#).

coretile.cluster0.l2_flusher
Type: AsyncCacheFlushUnit.

coretile.cluster1
ARM Cortex-A7 Cluster CT model.
Type: cluster_ARM_Cortex-A7.

coretile.cluster1.acp_mapper
Type: [PVBUSMapper](#).

coretile.cluster1.cpu0
ARM Cortex-A7 CT model.
Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu0.DTLB
Type: TLB.

coretile.cluster1.cpu0.ITLB
Type: TLB.

coretile.cluster1.cpu0.dtlb
TLB - instruction, data or unified.
Type: Tlbcadi.

coretile.cluster1.cpu0.itlb
TLB - instruction, data or unified.
Type: Tlbcadi.

coretile.cluster1.cpu0.l1dcache
PV Cache.
Type: pVCache.

coretile.cluster1.cpu0.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

coretile.cluster1.cpu0.l1icache
PV Cache.
Type: pVCache.

coretile.cluster1.cpu0.l1icache.upstream[0]
Type: [PVBUSSlave](#).

coretile.cluster1.cpu1

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).**coretile.cluster1.cpu1.DTLB**

Type: TLB.

coretile.cluster1.cpu1.ITLB

Type: TLB.

coretile.cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu1.l1dcache

PV Cache.

Type: pVCache.

coretile.cluster1.cpu1.l1dcache.upstream[0]Type: [PVBusSlave](#).**coretile.cluster1.cpu1.l1icache**

PV Cache.

Type: pVCache.

coretile.cluster1.cpu1.l1icache.upstream[0]Type: [PVBusSlave](#).**coretile.cluster1.cpu2**

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).**coretile.cluster1.cpu2.DTLB**

Type: TLB.

coretile.cluster1.cpu2.ITLB

Type: TLB.

coretile.cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu2.l1dcache

PV Cache.

Type: pVCache.

coretile.cluster1.cpu2.l1dcache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.cpu2.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.cpu3

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu3.DTLB

Type: [TLB](#).

coretile.cluster1.cpu3.ITLB

Type: [TLB](#).

coretile.cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster1.cpu3.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster1.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.ext_bus

Type: [PVBusLogger](#).

coretile.cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

coretile.cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[10]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[11]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[12]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[13]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[14]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[15]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[16]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[1]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[2]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[3]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[4]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[5]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[6]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

coretile.cluster1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

coretile.dualclustersystemconfigurationblock

Dual Cluster System Configuration Block.

Type: [DualClusterSystemConfigurationBlock](#).

coretile.dualclustersystemconfigurationblock.pvbuslave

Type: [PVBusSlave](#).

coretile.globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

coretile.globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

coretile.globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

coretile.v7_vgic

System VGIC architecture version v7.

Type: [v7_VGIC](#).

coretile.v7_vgic.vgic_bus_slave

Type: [PVBusSlave](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBUSMapper](#).

motherboard.flash0.mbs

Type: [PVBUSSlave](#).

motherboard.flash0.rmbs

Type: [PVBUSSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBUSMapper](#).

motherboard.flash1.mbs

Type: [PVBUSSlave](#).

motherboard.flash1.rmbs

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBusSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.psram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.smisc_91c111`

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

`motherboard.smisc_91c111.SMISC_slave`

Type: [PVBusSlave](#).

`motherboard.sp805_wdog`

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtiop9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtiop9device.mmio_slave`

Type: [PVBusSlave](#).

motherboard.virtio9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.7 FVP_VE_Cortex-A17x1

List of instances in FVP_VE_Cortex-A17x1.

FVP_VE_Cortex-A17x1 instances

cluster

ARM Cortex-A17 Cluster CT model.

Type: [Cluster_ARM_Cortex-A17](#).

cluster.acp_mapper

Type: [PVBusMapper](#).

cluster.cpu0

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).

cluster.cpu0.DTLB

Type: TLB.

cluster.cpu0.ITLB

Type: TLB.

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.ext_bus

Type: [PVBusLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.l2_cache

PV Cache.

Type: [PVCache](#).

cluster.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[7]

Type: [PVBUSSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBUSSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBUSSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBUSMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

gic400

GIC-400 Generic Interrupt Controller.

Type: [GIC_400](#).

gic400.vgic_bus_slave

Type: [PVBusSlave](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSlave](#).

motherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

`motherboard.pl180_mci.busslave`

Type: `PVBusSlave`.

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: `PS2Keyboard`.

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.psram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.smisc_91c111`

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

`motherboard.smc_91c111.SMSC_slave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog`

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

`motherboard.sp805_wdog.busslave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.sp805_wdog.clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

`motherboard.sp810_sysctrl.busslave`

Type: `PVBusSlave`.

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBUSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBUSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBUSSlave](#).

14.8 FVP_VE_Cortex-A17x1-A7x1

List of instances in FVP_VE_Cortex-A17x1-A7x1.

FVP_VE_Cortex-A17x1-A7x1 instances

coretile

Dual cluster ARM Cortex-A17x1 and ARM Cortex-A7x1 Core Tile.

Type: [ARM_Cortex_A17x1_A7x1_CT](#).

coretile.cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

coretile.cci400.cciinterconnect

Type: [PVCACHE](#).

coretile.cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIREGISTERS](#).

coretile.cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

coretile.cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

coretile.cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

coretile.cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

coretile.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.cluster0

ARM Cortex-A17 Cluster CT model.

Type: [Cluster_ARM_Cortex-A17](#).

coretile.cluster0.acp_mapper

Type: [PVBusMapper](#).

coretile.cluster0.cpu0

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).

coretile.cluster0.cpu0.DTLB

Type: TLB.

coretile.cluster0.cpu0.ITLB

Type: TLB.

coretile.cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

coretile.cluster0.cpu0.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

coretile.cluster0.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

coretile.cluster0.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.cpu0.l1icache

PV Cache.

Type: `PVCache`.

coretile.cluster0.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.ext_bus

Type: `PVBusLogger`.

coretile.cluster0.ext_bus.mapper

Type: `PVBusMapper`.

coretile.cluster0.l2_cache

PV Cache.

Type: `PVCache`.

coretile.cluster0.l2_cache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[10]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[11]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[12]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[13]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[14]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[15]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[16]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[1]

Type: `PVBusSlave`.

coretile.cluster0.l2_cache.upstream[2]Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[3]**Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[4]**Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[5]**Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[6]**Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[7]**Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[8]**Type: [PVBusSlave](#).**coretile.cluster0.l2_cache.upstream[9]**Type: [PVBusSlave](#).**coretile.cluster0.l2_flusher**Type: [AsyncCacheFlushUnit](#).**coretile.cluster1**

ARM Cortex-A7 Cluster CT model.

Type: [Cluster_ARM_Cortex-A7](#).**coretile.cluster1.acp_mapper**Type: [PVBusMapper](#).**coretile.cluster1.cpu0**

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).**coretile.cluster1.cpu0.DTLB**Type: [TLB](#).**coretile.cluster1.cpu0.ITLB**Type: [TLB](#).**coretile.cluster1.cpu0.dtlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**coretile.cluster1.cpu0.itlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**coretile.cluster1.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.ext_bus

Type: [PVBusLogger](#).

coretile.cluster1.ext_bus.mapper

Type: [PVBusMapper](#).

coretile.cluster1.l2_cache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.l2_cache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[10]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[11]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[12]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[13]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[14]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[15]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[16]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[1]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[2]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[3]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[4]

Type: [PVBusSlave](#).

coretile.cluster1.l2_cache.upstream[5]Type: [PVBUSSlave](#).**coretile.cluster1.l2_cache.upstream[6]**Type: [PVBUSSlave](#).**coretile.cluster1.l2_cache.upstream[7]**Type: [PVBUSSlave](#).**coretile.cluster1.l2_cache.upstream[8]**Type: [PVBUSSlave](#).**coretile.cluster1.l2_cache.upstream[9]**Type: [PVBUSSlave](#).**coretile.cluster1.l2_flusher**Type: [AsyncCacheFlushUnit](#).**coretile.dualclustersystemconfigurationblock**

Dual Cluster System Configuration Block.

Type: [DualClusterSystemConfigurationBlock](#).**coretile.dualclustersystemconfigurationblock.pvbusslave**Type: [PVBUSSlave](#).**coretile.globalcounter**

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).**coretile.globalcounter.pvbus_control_s[0]**Type: [PVBUSSlave](#).**coretile.globalcounter.pvbus_read_s[0]**Type: [PVBUSSlave](#).**coretile.v7_vgic**

System VGIC architecture version v7.

Type: [v7_VGIC](#).**coretile.v7_vgic.vgic_bus_slave**Type: [PVBUSSlave](#).**daughterboard**

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).**daughterboard.clockCLCD**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow [TrustZone](#) secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: `CounterModule`.

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: `AudioOut_SDL`.

`motherboard.clock100Hz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clock24MHz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clock35MHz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clock50Hz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clockCLCD`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`motherboard.dummy_local_dap_rom.bus_slave`

Type: `PVBusSlave`.

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSlave](#).

motherboard.pl011_uart0.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBUSlave](#).

`motherboard.pl011_uart1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart2.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart2.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi1.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl111_clcd`

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd`

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd.busmaster`

Type: [PVBusMaster](#).

`motherboard.pl111_clcd.pl11x_clcd.busslave`

Type: [PVBusSlave](#).

`motherboard.pl111_clcd.pl11x_clcd.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

`motherboard.sp810_sysctrl.busslave`

Type: [PVBusSlave](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [VE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBUSSlave](#).

motherboard.virtio_blockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtio_blockdevice.register_slave

Type: [PVBUSSlave](#).

motherboard.virtio_blockdevice.virtio_master

Type: [PVBUSMaster](#).

motherboard.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtio_p9device.mmio_slave

Type: [PVBUSSlave](#).

motherboard.virtio_p9device.virtio_master

Type: [PVBUSMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBUSSlave](#).

14.9 FVP_VE_Cortex-A17x2

List of instances in FVP_VE_Cortex-A17x2.

FVP_VE_Cortex-A17x2 instances

cluster

ARM Cortex-A17 Cluster CT model.

Type: [Cluster_ARM_Cortex-A17](#).

cluster.acp_mapper

Type: [PVBUSMapper](#).

cluster.cpu0

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).

cluster.cpu0.DTLB

Type: TLB.

cluster.cpu0.ITLB

Type: TLB.

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu0.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu0.l1icache

PV Cache.

Type: pVCache.

cluster.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

cluster.cpu1

ARM Cortex-A17 CT model.

Type: ARM_Cortex-A17.

cluster.cpu1.DTLB

Type: TLB.

cluster.cpu1.ITLB

Type: TLB.

cluster.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

cluster.cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.ext_bus

Type: [PVBusLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.l2_cache

PV Cache.

Type: [PVCache](#).

cluster.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

gic400

GIC-400 Generic Interrupt Controller.

Type: [GIC_400](#).

gic400.vgic_bus_slave

Type: [PVBusSlave](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

`motherboard.Timer_0_1.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_0_1.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3`

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

`motherboard.Timer_2_3.busslave`

Type: [PVBusSlave](#).

`motherboard.Timer_2_3.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_local_dap_rom.bus_slave

Type: [PVBUSSlave](#).

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBusSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBusMapper](#).

`motherboard.flash0.mbs`

Type: [PVBusSlave](#).

`motherboard.flash0.rmbs`

Type: [PVBusSlave](#).

`motherboard.flash1`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash1.map`

Type: [PVBusMapper](#).

`motherboard.flash1.mbs`

Type: [PVBusSlave](#).

`motherboard.flash1.rmbs`

Type: [PVBusSlave](#).

`motherboard.flashloader0`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.flashloader1`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.hostbridge`

Host Socket Interface Component.

Type: [HostBridge](#).

`motherboard.mmc`

Generic Multimedia Card.

Type: [MMC](#).

`motherboard.mmc.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBusSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl031_rtc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl031_rtc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi1.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl111_clcd`

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd`

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd.busmaster`

Type: [PVBUSMaster](#).

`motherboard.pl111_clcd.pl11x_clcd.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl111_clcd.pl11x_clcd.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBUSSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBUSSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [vE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.10 FVP_VE_Cortex-A17x4

List of instances in FVP_VE_Cortex-A17x4.

FVP_VE_Cortex-A17x4 instances

cluster

ARM Cortex-A17 Cluster CT model.

Type: [Cluster_ARM_Cortex-A17](#).

cluster.acp_mapper

Type: [PVBusMapper](#).

cluster.cpu0

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).

cluster.cpu0.DTLB

Type: TLB.

cluster.cpu0.ITLB

Type: TLB.

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster.cpu0.l1dcache**

PV Cache.

Type: `PvCache`.**cluster.cpu0.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu0.l1icache**

PV Cache.

Type: `PvCache`.**cluster.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu1**

ARM Cortex-A17 CT model.

Type: `ARM_Cortex-A17`.**cluster.cpu1.DTLB**Type: `TLB`.**cluster.cpu1.ITLB**Type: `TLB`.**cluster.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster.cpu1.itlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster.cpu1.l1dcache**

PV Cache.

Type: `PvCache`.**cluster.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu1.l1icache**

PV Cache.

Type: `PvCache`.**cluster.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu2**

ARM Cortex-A17 CT model.

Type: `ARM_Cortex-A17`.

cluster.cpu2.DTLB

Type: TLB.

cluster.cpu2.ITLB

Type: TLB.

cluster.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu2.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu2.l1icache

PV Cache.

Type: pVCache.

cluster.cpu2.l1icache.upstream[0]

Type: PVBusSlave.

cluster.cpu3

ARM Cortex-A17 CT model.

Type: ARM_Cortex-A17.

cluster.cpu3.DTLB

Type: TLB.

cluster.cpu3.ITLB

Type: TLB.

cluster.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu3.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu3.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu3.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

cluster.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.ext_bus

Type: [PVBusLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.l2_cache

PV Cache.

Type: [PVCache](#).

cluster.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

gic400

GIC-400 Generic Interrupt Controller.

Type: [GIC_400](#).

gic400.vgic_bus_slave

Type: [PVBusSlave](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

`motherboard.Timer_0_1.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_0_1.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3`

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

`motherboard.Timer_2_3.busslave`

Type: [PVBusSlave](#).

`motherboard.Timer_2_3.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_local_dap_rom.bus_slave

Type: [PVBUSSlave](#).

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBUSSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.mmc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBusSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBUSSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [vE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.11 FVP_VE_Cortex-A17x4-A7x4

List of instances in FVP_VE_Cortex-A17x4-A7x4.

FVP_VE_Cortex-A17x4-A7x4 instances

coretile

Dual cluster ARM Cortex-A17x4 and ARM Cortex-A7x4 Core Tile.

Type: [ARM_Cortex_A17x4_A7x4_CT](#).

coretile.cci400

Cache Coherent Interconnect for AXI4 ACE.

Type: [CCI400](#).

coretile.cci400.cciinterconnect

Type: [PVCache](#).

coretile.cci400.cciregisters

Programmer-visible memory mapped registers for use by CCI400.

Type: [CCIRegisters](#).

coretile.cci400.cciregisters.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

coretile.cci400.cciregisters.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

coretile.cci400.cciregisters.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

coretile.cci400.cciregisters.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

coretile.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.clockdivider1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

coretile.cluster0

ARM Cortex-A17 Cluster CT model.

Type: [Cluster_ARM_Cortex-A17](#).

coretile.cluster0.acp_mapper

Type: [PVBusMapper](#).

coretile.cluster0.cpu0

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).

coretile.cluster0.cpu0.DTLB

Type: [TLB](#).

coretile.cluster0.cpu0.ITLB

Type: [TLB](#).

coretile.cluster0.cpu0.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu0.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu0.l1dcache

PV Cache.

Type: pVCache.

coretile.cluster0.cpu0.l1dcache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu0.l1icache

PV Cache.

Type: pVCache.

coretile.cluster0.cpu0.l1icache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu1

ARM Cortex-A17 CT model.

Type: ARM_Cortex-A17.

coretile.cluster0.cpu1.DTLB

Type: TLB.

coretile.cluster0.cpu1.ITLB

Type: TLB.

coretile.cluster0.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu1.l1dcache

PV Cache.

Type: pVCache.

coretile.cluster0.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu1.l1icache

PV Cache.

Type: pVCache.

coretile.cluster0.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

coretile.cluster0.cpu2

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).**coretile.cluster0.cpu2.DTLB**

Type: TLB.

coretile.cluster0.cpu2.ITLB

Type: TLB.

coretile.cluster0.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu2.l1dcache

PV Cache.

Type: pVCache.

coretile.cluster0.cpu2.l1dcache.upstream[0]Type: [PVBUSSlave](#).**coretile.cluster0.cpu2.l1icache**

PV Cache.

Type: pVCache.

coretile.cluster0.cpu2.l1icache.upstream[0]Type: [PVBUSSlave](#).**coretile.cluster0.cpu3**

ARM Cortex-A17 CT model.

Type: [ARM_Cortex-A17](#).**coretile.cluster0.cpu3.DTLB**

Type: TLB.

coretile.cluster0.cpu3.ITLB

Type: TLB.

coretile.cluster0.cpu3.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu3.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster0.cpu3.l1dcache

PV Cache.

Type: pVCache.

coretile.cluster0.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster0.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.ext_bus

Type: [PVBusLogger](#).

coretile.cluster0.ext_bus.mapper

Type: [PVBusMapper](#).

coretile.cluster0.l2_cache

PV Cache.

Type: [PVCache](#).

coretile.cluster0.l2_cache.upstream[0]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[10]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[11]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[12]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[13]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[14]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[15]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[16]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[1]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[2]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[3]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[4]

Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[5]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[6]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[7]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[8]
Type: [PVBusSlave](#).

coretile.cluster0.l2_cache.upstream[9]
Type: [PVBusSlave](#).

coretile.cluster0.l2_flusher
Type: AsyncCacheFlushUnit.

coretile.cluster1
ARM Cortex-A7 Cluster CT model.
Type: Cluster_ARM_Cortex-A7.

coretile.cluster1.acp_mapper
Type: [PVBusMapper](#).

coretile.cluster1.cpu0
ARM Cortex-A7 CT model.
Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu0.DTLB
Type: TLB.

coretile.cluster1.cpu0.ITLB
Type: TLB.

coretile.cluster1.cpu0.dtlb
TLB - instruction, data or unified.
Type: TlbCadi.

coretile.cluster1.cpu0.itlb
TLB - instruction, data or unified.
Type: TlbCadi.

coretile.cluster1.cpu0.l1dcache
PV Cache.
Type: [PVCache](#).

coretile.cluster1.cpu0.l1dcache.upstream[0]
Type: [PVBusSlave](#).

coretile.cluster1.cpu0.l1icache
PV Cache.
Type: [PVCache](#).

coretile.cluster1.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

coretile.cluster1.cpu1

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu1.DTLB

Type: TLB.

coretile.cluster1.cpu1.ITLB

Type: TLB.

coretile.cluster1.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu1.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

coretile.cluster1.cpu1.l1icache

PV Cache.

Type: [PVCache](#).

coretile.cluster1.cpu1.l1icache.upstream[0]

Type: [PVBUSSlave](#).

coretile.cluster1.cpu2

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

coretile.cluster1.cpu2.DTLB

Type: TLB.

coretile.cluster1.cpu2.ITLB

Type: TLB.

coretile.cluster1.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

coretile.cluster1.cpu2.l1dcache

PV Cache.

Type: `PVCache`.

coretile.cluster1.cpu2.l1dcache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster1.cpu2.l1icache

PV Cache.

Type: `PVCache`.

coretile.cluster1.cpu2.l1icache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster1.cpu3

ARM Cortex-A7 CT model.

Type: `ARM_Cortex-A7`.

coretile.cluster1.cpu3.DTLB

Type: `TLB`.

coretile.cluster1.cpu3.ITLB

Type: `TLB`.

coretile.cluster1.cpu3.dtlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

coretile.cluster1.cpu3.itlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

coretile.cluster1.cpu3.l1dcache

PV Cache.

Type: `PVCache`.

coretile.cluster1.cpu3.l1dcache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster1.cpu3.l1icache

PV Cache.

Type: `PVCache`.

coretile.cluster1.cpu3.l1icache.upstream[0]

Type: `PVBusSlave`.

coretile.cluster1.ext_bus

Type: `PVBusLogger`.

coretile.cluster1.ext_bus.mapper

Type: `PVBusMapper`.

coretile.cluster1.l2_cache

PV Cache.

Type: `PVCache`.

`coretile.cluster1.l2_cache.upstream[0]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[10]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[11]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[12]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[13]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[14]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[15]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[16]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[1]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[2]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[3]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[4]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[5]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[6]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[7]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[8]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_cache.upstream[9]`
Type: `PVBusSlave`.

`coretile.cluster1.l2_flusher`
Type: `AsyncCacheFlushUnit`.

coretile.dualclustersystemconfigurationblock

Dual Cluster System Configuration Block.

Type: [DualClusterSystemConfigurationBlock](#).

coretile.dualclustersystemconfigurationblock.pvbusslave

Type: [PVBusSlave](#).

coretile.globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

coretile.globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

coretile.globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

coretile.v7_vgic

System VGIC architecture version v7.

Type: [v7_VGIC](#).

coretile.v7_vgic.vgic_bus_slave

Type: [PVBusSlave](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmb

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_0_1.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_0_1.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3`

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

`motherboard.Timer_2_3.busslave`

Type: [PVBusSlave](#).

`motherboard.Timer_2_3.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

`motherboard.clock100Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [`SchedulerThread`](#).

`motherboard.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [`SchedulerThreadEvent`](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [`PL011_Uart`](#).

`motherboard.pl011_uart0.busslave`

Type: [`PVBusSlave`](#).

`motherboard.pl011_uart0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [`ClockDivider`](#).

`motherboard.pl011_uart0.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread`](#).

`motherboard.pl011_uart0.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

`motherboard.pl180_mci.busslave`

Type: `PVBusSlave`.

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: `PS2Keyboard`.

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.psram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.smisc_91c111`

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

`motherboard.smc_91c111.SMSC_slave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog`

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

`motherboard.sp805_wdog.busslave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.sp805_wdog.clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

`motherboard.sp810_sysctrl.busslave`

Type: `PVBusSlave`.

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBUSlave](#).

`motherboard.virtioBlockDevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioBlockDevice.register_slave`

Type: [PVBUSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBUSSlave](#).

14.12 FVP_VE_Cortex-A5x1

List of instances in FVP_VE_Cortex-A5x1.

FVP_VE_Cortex-A5x1 instances

cluster

ARM CORTEXA5MP Cluster CT model.

Type: [Cluster_ARM_Cortex-A5MP](#).

cluster.acp_mapper

Type: [PVBUSMapper](#).

cluster.cpu0

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).

cluster.cpu0.UTLB

Type: [TLB](#).

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster.cpu0.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu0.utlb

TLB - instruction, data or unified.

Type: [Tlbcadi](#).

cluster.ext_bus

Type: [PVBusLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.internal_shareability_remapper

Type: [PVBusMapper](#).

cluster.l1_incoherent_interconnect

Type: [PVCache](#).

cluster.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[2]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[3]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

cluster.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

cluster.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

cluster.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

cluster.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cluster.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBUSMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBUSMapper](#).

motherboard.flash0.mbs

Type: [PVBUSSlave](#).

motherboard.flash0.rmbs

Type: [PVBUSSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBUSMapper](#).

motherboard.flash1.mbs

Type: [PVBUSSlave](#).

motherboard.flash1.rmbs

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

motherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtiop9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtiop9device.mmio_slave`

Type: [PVBusSlave](#).

motherboard.virtio9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.13 FVP_VE_Cortex-A5x2

List of instances in FVP_VE_Cortex-A5x2.

FVP_VE_Cortex-A5x2 instances

cluster

ARM CORTEXA5MP Cluster CT model.

Type: [Cluster_ARM_Cortex-A5MP](#).

cluster.acp_mapper

Type: [PVBusMapper](#).

cluster.cpu0

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).

cluster.cpu0.UTLB

Type: [TLB](#).

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu0.utlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).**cluster.cpu1.UTLB**

Type: TLB.

cluster.cpu1.l1dcache

PV Cache.

Type: PVCache.

cluster.cpu1.l1dcache.upstream[0]Type: [PVBusSlave](#).**cluster.cpu1.l1icache**

PV Cache.

Type: PVCache.

cluster.cpu1.l1icache.upstream[0]Type: [PVBusSlave](#).**cluster.cpu1.utlb**

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.ext_busType: [PVBusLogger](#).**cluster.ext_bus.mapper**Type: [PVBusMapper](#).**cluster.internal_shareability_remapper**Type: [PVBusMapper](#).**cluster.l1_incoherent_interconnect**

Type: PVCache.

cluster.l1_incoherent_interconnect.upstream[0]Type: [PVBusSlave](#).**cluster.l1_incoherent_interconnect.upstream[10]**Type: [PVBusSlave](#).**cluster.l1_incoherent_interconnect.upstream[11]**Type: [PVBusSlave](#).**cluster.l1_incoherent_interconnect.upstream[12]**Type: [PVBusSlave](#).**cluster.l1_incoherent_interconnect.upstream[13]**Type: [PVBusSlave](#).

cluster.11_incoherent_interconnect.upstream[14]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[15]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cluster.12_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow [TrustZone](#) secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: `CounterModule`.

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: `AudioOut_SDL`.

`motherboard.clock100Hz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clock24MHz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clock35MHz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clock50Hz`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.clockCLCD`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`motherboard.dummy_local_dap_rom.bus_slave`

Type: `PVBusSlave`.

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

`motherboard.mmc.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBUSlave](#).

`motherboard.pl011_uart0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBUSlave](#).

`motherboard.pl011_uart1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart2.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart2.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBusSlave](#).

motherboard.pl031_rtc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBusSlave](#).

motherboard.pl041_aaci.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi1.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl111_clcd`

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd`

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

`motherboard.pl111_clcd.pl11x_clcd.busmaster`

Type: [PVBusMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

motherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

motherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

`motherboard.sp810_sysctrl.busslave`

Type: [PVBusSlave](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [VE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBUSSlave](#).

motherboard.virtio_blockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtio_blockdevice.register_slave

Type: [PVBUSSlave](#).

motherboard.virtio_blockdevice.virtio_master

Type: [PVBUSMaster](#).

motherboard.virtio_p9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtio_p9device.mmio_slave

Type: [PVBUSSlave](#).

motherboard.virtio_p9device.virtio_master

Type: [PVBUSMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBUSSlave](#).

periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.14 FVP_VE_Cortex-A5x4

List of instances in FVP_VE_Cortex-A5x4.

FVP_VE_Cortex-A5x4 instances

cluster

ARM CORTEXA5MP Cluster CT model.

Type: [Cluster_ARM_Cortex-A5MP](#).

cluster.acp_mapperType: [PVBusMapper](#).**cluster.cpu0**

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).**cluster.cpu0.UTLB**

Type: TLB.

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).**cluster.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**cluster.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster.cpu0.utlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster.cpu1**

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).**cluster.cpu1.UTLB**

Type: TLB.

cluster.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).**cluster.cpu1.l1dcache.upstream[0]**Type: [PVBusSlave](#).**cluster.cpu1.l1icache**

PV Cache.

Type: [PVCache](#).**cluster.cpu1.l1icache.upstream[0]**Type: [PVBusSlave](#).**cluster.cpu1.utlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu2

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).**cluster.cpu2.UTLB**

Type: TLB.

cluster.cpu2.l1dcache

PV Cache.

Type: [pVCache](#).**cluster.cpu2.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster.cpu2.l1icache**

PV Cache.

Type: [pVCache](#).**cluster.cpu2.l1icache.upstream[0]**Type: [PVBUSSlave](#).**cluster.cpu2.utlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster.cpu3**

ARM CORTEXA5MP CT model.

Type: [ARM_Cortex-A5MP](#).**cluster.cpu3.UTLB**

Type: TLB.

cluster.cpu3.l1dcache

PV Cache.

Type: [pVCache](#).**cluster.cpu3.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**cluster.cpu3.l1icache**

PV Cache.

Type: [pVCache](#).**cluster.cpu3.l1icache.upstream[0]**Type: [PVBUSSlave](#).**cluster.cpu3.utlb**

TLB - instruction, data or unified.

Type: [TlbCadi](#).**cluster.ext_bus**Type: [PVBUSLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.internal_shareability_remapper

Type: [PVBusMapper](#).

cluster.l1_incoherent_interconnect

Type: [PVCache](#).

cluster.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[2]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[3]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[4]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[5]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[6]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[7]

Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cluster.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBUSMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBUSMapper](#).

motherboard.flash1.mbs

Type: [PVBUSSlave](#).

motherboard.flash1.rmbs

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart2.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart2.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBusSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBusSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBusSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.sp805_wdog.clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [`SchedulerThread`](#).

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [`SchedulerThreadEvent`](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller (SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [`SP810_SysCtrl`](#).

`motherboard.sp810_sysctrl.busslave`

Type: [`PVBusSlave`](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [`ClockDivider`](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [`ClockDivider`](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [`ClockDivider`](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtio_blockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtio_blockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtio_blockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtio_p9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtio_p9device.mmio_slave`

Type: [PVBusSlave](#).

`motherboard.virtio_p9device.virtio_master`

Type: [PVBusMaster](#).

`motherboard.vis`

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

`motherboard.vis.recorder`

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

`motherboard.vis.recorder.playbackDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vis.recorder.playbackTimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.vis.recorder.playbackTimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.vis.recorder.playbackTimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.vis.recorder.recordingDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.vram.bus_slave`

Type: [PVBUSSlave](#).

periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.15 FVP_VE_Cortex-A7x1

List of instances in FVP_VE_Cortex-A7x1.

FVP_VE_Cortex-A7x1 instances

cluster

ARM Cortex-A7 Cluster CT model.

Type: [cluster_ARM_Cortex-A7](#).

cluster.Cortex-A7_GIC

Type: [GICv2](#).

cluster.acp_mapper

Type: [PVBusMapper](#).

cluster.cpu0

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

cluster.cpu0.DTLB

Type: [TLB](#).

cluster.cpu0.ITLB

Type: [TLB](#).

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.ext_bus

Type: [PVBusLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.l2_cache

PV Cache.

Type: [PVCache](#).

cluster.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[12]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[13]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[14]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[15]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[16]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[1]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[2]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[3]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[4]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[5]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[6]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[7]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[8]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[9]

Type: [PVBusSlave](#).

cluster.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBUSSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBUSMapper](#).

motherboard.flash0.mbs

Type: [PVBUSSlave](#).

motherboard.flash0.rmbs

Type: [PVBUSSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBUSMapper](#).

motherboard.flash1.mbs

Type: [PVBUSSlave](#).

motherboard.flash1.rmbs

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBusSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBusSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

`motherboard.pl180_mci.busslave`

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

motherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smisc_91c111.SMISC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBusSlave](#).

`motherboard.virtioblockdevice.virtio_master`

Type: [PVBusMaster](#).

`motherboard.virtiop9device`

virtio P9 server.

Type: [VirtioP9Device](#).

`motherboard.virtiop9device.mmio_slave`

Type: [PVBusSlave](#).

motherboard.virtio9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBusSlave](#).

14.16 FVP_VE_Cortex-A7x2

List of instances in FVP_VE_Cortex-A7x2.

FVP_VE_Cortex-A7x2 instances

cluster

ARM Cortex-A7 Cluster CT model.

Type: [Cluster_ARM_Cortex-A7](#).

cluster.Cortex-A7_GIC

Type: [GICv2](#).

cluster.acp_mapper

Type: [PVBusMapper](#).

cluster.cpu0

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

cluster.cpu0.DTLB

Type: [TLB](#).

cluster.cpu0.ITLB

Type: [TLB](#).

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

cluster.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu0.l1icache

PV Cache.

Type: `PVCache`.**cluster.cpu0.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu1**

ARM Cortex-A7 CT model.

Type: `ARM_Cortex-A7`.**cluster.cpu1.DTLB**Type: `TLB`.**cluster.cpu1.ITLB**Type: `TLB`.**cluster.cpu1.dtlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster.cpu1.itlb**

TLB - instruction, data or unified.

Type: `Tlbcadi`.**cluster.cpu1.l1dcache**

PV Cache.

Type: `PVCache`.**cluster.cpu1.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu1.l1icache**

PV Cache.

Type: `PVCache`.**cluster.cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster.ext_bus**Type: `PVBusLogger`.**cluster.ext_bus.mapper**Type: `PVBusMapper`.**cluster.l2_cache**

PV Cache.

Type: `PVCache`.**cluster.l2_cache.upstream[0]**Type: `PVBusSlave`.**cluster.l2_cache.upstream[10]**Type: `PVBusSlave`.

cluster.l2_cache.upstream[11]Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[12]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[13]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[14]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[15]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[16]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[1]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[2]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[3]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[4]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[5]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[6]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[7]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[8]**Type: [PVBUSSlave](#).**cluster.l2_cache.upstream[9]**Type: [PVBUSSlave](#).**cluster.l2_flusher**Type: [AsyncCacheFlushUnit](#).**daughterboard**

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).**daughterboard.clockCLCD**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapperType: [PVBusMapper](#).**daughterboard.exclusive_monitor**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).**daughterboard.exclusive_monitor.bus_mapper**Type: [PVBusMapper](#).**daughterboard.hdlcd**

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCDC](#).**daughterboard.hdlcd.busmaster**Type: [PVBusMaster](#).**daughterboard.hdlcd.busslave**Type: [PVBusSlave](#).**daughterboard.hdlcd.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**daughterboard.hdlcd.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**daughterboard.hdlcd.timer.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**daughterboard.hdlcd.timer.timer.thread_event**A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.Type: [SchedulerThreadEvent](#).**daughterboard.introuter**

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: `VEDCC`.

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: `MemoryMappedCounterModule`.

globalcounter.pvbus_control_s[0]

Type: `PVBusSlave`.

globalcounter.pvbus_read_s[0]

Type: `PVBusSlave`.

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: `VEMotherBoard`.

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

motherboard.Timer_0_1.busslave

Type: `PVBusSlave`.

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: `SP804_Timer`.

motherboard.Timer_2_3.busslave

Type: `PVBusSlave`.

`motherboard.Timer_2_3.clk_div0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

`motherboard.clock100Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBusSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBusMapper](#).

`motherboard.flash0.mbs`

Type: [PVBusSlave](#).

`motherboard.flash0.rmbs`

Type: [PVBusSlave](#).

`motherboard.flash1`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash1.map`

Type: [PVBusMapper](#).

`motherboard.flash1.mbs`

Type: [PVBusSlave](#).

motherboard.flash1.rmb

Type: [PVBUSSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

motherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.pl011_uart1.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl011_uart1.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl011_uart1.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`motherboard.pl011_uart2.busslave`

Type: `PVBusSlave`.

`motherboard.pl011_uart2.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.pl011_uart2.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.pl011_uart2.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl011_uart2.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl011_uart2.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`motherboard.pl011_uart3.busslave`

Type: `PVBusSlave`.

`motherboard.pl011_uart3.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.pl011_uart3.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.pl011_uart3.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl031_rtc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl031_rtc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.pl041_aaci.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.pl041_aaci.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.pl041_aaci.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: `PL050_KMI`.

`motherboard.pl050_kmi0.busslave`

Type: `PVBusSlave`.

`motherboard.pl050_kmi0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: `PL050_KMI`.

`motherboard.pl050_kmi1.busslave`

Type: `PVBusSlave`.

`motherboard.pl050_kmi1.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBUSSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

`motherboard.psram.bus_slave`

Type: `PVBusSlave`.

`motherboard.smsc_91c111`

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

`motherboard.smsc_91c111.SMSC_slave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog`

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

`motherboard.sp805_wdog.busslave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.sp805_wdog.clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

`motherboard.sp810_sysctrl.busslave`

Type: [PVBusSlave](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [VE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vis.recorder.playbackTimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.vis.recorder.playbackTimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.vis.recorder.playbackTimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.vis.recorder.recordingDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.vram.bus_slave`

Type: [PVBusSlave](#).

14.17 FVP_VE_Cortex-A7x4

List of instances in FVP_VE_Cortex-A7x4.

FVP_VE_Cortex-A7x4 instances

cluster

ARM Cortex-A7 Cluster CT model.

Type: `cluster_ARM_Cortex-A7`.

cluster.Cortex-A7_GIC

Type: `GICv2`.

cluster.acp_mapper

Type: `PVBusMapper`.

cluster.cpu0

ARM Cortex-A7 CT model.

Type: `ARM_Cortex-A7`.

cluster.cpu0.DTLB

Type: `TLB`.

cluster.cpu0.ITLB

Type: `TLB`.

cluster.cpu0.dtlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster.cpu0.itlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu1

ARM Cortex-A7 CT model.

Type: `ARM_Cortex-A7`.

cluster.cpu1.DTLB

Type: `TLB`.

cluster.cpu1.ITLB

Type: TLB.

cluster.cpu1.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu1.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu1.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu1.l1icache

PV Cache.

Type: pVCache.

cluster.cpu1.l1icache.upstream[0]

Type: PVBusSlave.

cluster.cpu2

ARM Cortex-A7 CT model.

Type: ARM_Cortex-A7.

cluster.cpu2.DTLB

Type: TLB.

cluster.cpu2.ITLB

Type: TLB.

cluster.cpu2.dtlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu2.itlb

TLB - instruction, data or unified.

Type: TlbCadi.

cluster.cpu2.l1dcache

PV Cache.

Type: pVCache.

cluster.cpu2.l1dcache.upstream[0]

Type: PVBusSlave.

cluster.cpu2.l1icache

PV Cache.

Type: pVCache.

cluster.cpu2.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu3

ARM Cortex-A7 CT model.

Type: [ARM_Cortex-A7](#).

cluster.cpu3.DTLB

Type: TLB.

cluster.cpu3.ITLB

Type: TLB.

cluster.cpu3.dtlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu3.itlb

TLB - instruction, data or unified.

Type: [TlbCadi](#).

cluster.cpu3.l1dcache

PV Cache.

Type: [pVCache](#).

cluster.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cluster.cpu3.l1icache

PV Cache.

Type: [pVCache](#).

cluster.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

cluster.ext_bus

Type: [PVBusLogger](#).

cluster.ext_bus.mapper

Type: [PVBusMapper](#).

cluster.l2_cache

PV Cache.

Type: [pVCache](#).

cluster.l2_cache.upstream[0]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[10]

Type: [PVBusSlave](#).

cluster.l2_cache.upstream[11]

Type: [PVBusSlave](#).

cluster.12_cache.upstream[12]Type: [PVBUSSlave](#).**cluster.12_cache.upstream[13]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[14]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[15]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[16]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[1]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[2]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[3]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[4]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[5]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[6]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[7]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[8]**Type: [PVBUSSlave](#).**cluster.12_cache.upstream[9]**Type: [PVBUSSlave](#).**cluster.12_flusher**Type: [AsyncCacheFlushUnit](#).**daughterboard**

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).**daughterboard.clockCLCD**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow [TrustZone](#) secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

globalcounter

Memory Mapped Counter Module for Generic Timers.

Type: [MemoryMappedCounterModule](#).

globalcounter.pvbus_control_s[0]

Type: [PVBusSlave](#).

globalcounter.pvbus_read_s[0]

Type: [PVBusSlave](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

`motherboard.clock100Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBusSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBusMapper](#).

`motherboard.flash0.mbs`

Type: [PVBusSlave](#).

`motherboard.flash0.rmbs`

Type: [PVBusSlave](#).

`motherboard.flash1`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash1.map`

Type: [PVBusMapper](#).

`motherboard.flash1.mbs`

Type: [PVBusSlave](#).

`motherboard.flash1.rmbs`

Type: [PVBusSlave](#).

`motherboard.flashloader0`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.flashloader1`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.hostbridge`

Host Socket Interface Component.

Type: [HostBridge](#).

`motherboard.mmc`

Generic Multimedia Card.

Type: [MMC](#).

`motherboard.mmc.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart1.busslave

Type: [PVBUSlave](#).

motherboard.pl011_uart1.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart2.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart2.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl031_rtc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl031_rtc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

motherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

motherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

motherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBUSSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

`motherboard.sp810_sysctrl.busslave`

Type: [PVBusSlave](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [VE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vis.recorder.playbackTimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.vis.recorder.playbackTimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.vis.recorder.playbackTimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.vis.recorder.recordingDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.vram.bus_slave`

Type: [PVBusSlave](#).

14.18 FVP_VE_Cortex-A9x1

List of instances in FVP_VE_Cortex-A9x1.

FVP_VE_Cortex-A9x1 instances

cluster

ARM CORTEXA9MP Cluster CT model.

Type: `Cluster_ARM_Cortex-A9MP`.

cluster.acp_mapper

Type: `PVBusMapper`.

cluster.cpu0

ARM CORTEXA9MP CT model.

Type: `ARM_Cortex-A9MP`.

cluster.cpu0.UTLB

Type: `TLB`.

cluster.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.utlb

TLB - instruction, data or unified.

Type: `Tlbcadi`.

cluster.ext_bus

Type: `PVBusLogger`.

cluster.ext_bus.mapper

Type: `PVBusMapper`.

cluster.internal_shareability_remapper

Type: `PVBusMapper`.

cluster.l1_incoherent_interconnect

Type: `PVCache`.

cluster.l1_incoherent_interconnect.upstream[0]

Type: `PVBusSlave`.

cluster.11_incoherent_interconnect.upstream[10]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[11]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[12]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[13]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[14]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[15]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cluster.12_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: `VEDaughterBoard`.

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

daughterboard.coresight_mapper

Type: `PVBusMapper`.

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

daughterboard.dmc.bus_slave

Type: `PVBusSlave`.

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

daughterboard.dmc_phy.bus_slave

Type: `PVBusSlave`.

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

daughterboard.dram.bus_slave

Type: `PVBusSlave`.

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: `TZSwitch`.

daughterboard.dram_aliased.pvbus_mapper

Type: `PVBusMapper`.

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: `TZSwitch`.

daughterboard.dram_limit_4.pvbus_mapper

Type: `PVBusMapper`.

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

`motherboard.clock100Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.dummy_local_dap_rom

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_local_dap_rom.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

motherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This means that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBUSlave](#).

`motherboard.pl011_uart1.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart2`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart2.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart2.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart2.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart2.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

motherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

motherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi1.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

motherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

motherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

motherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBusSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

motherboard.sp810_sysctrl.busslave

Type: [PVBusSlave](#).

motherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [VE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vis.recorder.playbackTimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.vis.recorder.playbackTimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.vis.recorder.playbackTimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.vis.recorder.recordingDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.vram.bus_slave`

Type: [PVBusSlave](#).

`periph_clockdivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

14.19 FVP_VE_Cortex-A9x2

List of instances in FVP_VE_Cortex-A9x2.

FVP_VE_Cortex-A9x2 instances

cluster

ARM CORTEXA9MP Cluster CT model.

Type: `Cluster_ARM_Cortex-A9MP`.

cluster.acp_mapper

Type: `PVBusMapper`.

cluster.cpu0

ARM CORTEXA9MP CT model.

Type: `ARM_Cortex-A9MP`.

cluster.cpu0.UTLB

Type: `TLB`.

cluster.cpu0.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.l1icache

PV Cache.

Type: `PVCache`.

cluster.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

cluster.cpu0.utlb

TLB - instruction, data or unified.

Type: `TlbCadi`.

cluster.cpu1

ARM CORTEXA9MP CT model.

Type: `ARM_Cortex-A9MP`.

cluster.cpu1.UTLB

Type: `TLB`.

cluster.cpu1.l1dcache

PV Cache.

Type: `PVCache`.

cluster.cpu1.l1dcache.upstream[0]
Type: [PVBusSlave](#).

cluster.cpu1.l1icache
PV Cache.
Type: [PVCache](#).

cluster.cpu1.l1icache.upstream[0]
Type: [PVBusSlave](#).

cluster.cpu1.utlb
TLB - instruction, data or unified.
Type: [TlbCadi](#).

cluster.ext_bus
Type: [PVBusLogger](#).

cluster.ext_bus.mapper
Type: [PVBusMapper](#).

cluster.internal_shareability_remapper
Type: [PVBusMapper](#).

cluster.l1_incoherent_interconnect
Type: [PVCache](#).

cluster.l1_incoherent_interconnect.upstream[0]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[10]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[11]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[12]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[13]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[14]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[15]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[16]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[17]
Type: [PVBusSlave](#).

cluster.l1_incoherent_interconnect.upstream[1]
Type: [PVBusSlave](#).

cluster.11_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cluster.12_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBUSMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBUSSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmb

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.Timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

motherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

motherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBUSSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBUSSlave](#).

motherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash0.map

Type: [PVBusMapper](#).

motherboard.flash0.mbs

Type: [PVBusSlave](#).

motherboard.flash0.rmbs

Type: [PVBusSlave](#).

motherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

motherboard.flash1.map

Type: [PVBusMapper](#).

motherboard.flash1.mbs

Type: [PVBusSlave](#).

motherboard.flash1.rmbs

Type: [PVBusSlave](#).

motherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

motherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

motherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

motherboard.mmc.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBusSlave](#).

`motherboard.pl011_uart1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart1.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart1.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart1.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

motherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

motherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

motherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

motherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

motherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [`SchedulerThread`](#).

`motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [`SchedulerThreadEvent`](#).

`motherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [`PL180_MCI`](#).

`motherboard.pl180_mci.busslave`

Type: [`PVBusSlave`](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: [`PS2Keyboard`](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread`](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [`ClockTimerThread64`](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2mouse.ps2_clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.psram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.psram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.smisc_91c111`

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

`motherboard.smc_91c111.SMSC_slave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog`

ARM Watchdog Module(SP805).

Type: `SP805_Watchdog`.

`motherboard.sp805_wdog.busslave`

Type: `PVBusSlave`.

`motherboard.sp805_wdog.clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.sp805_wdog.clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.sp805_wdog.clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: `SP810_SysCtrl`.

`motherboard.sp810_sysctrl.busslave`

Type: `PVBusSlave`.

`motherboard.sp810_sysctrl.clkdiv_clk0`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.terminal_0`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_1`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_2`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.terminal_3`

Telnet terminal interface.

Type: [TelnetTerminal](#).

`motherboard.ve_sysregs`

Type: [VE_SysRegs](#).

`motherboard.ve_sysregs.busslave`

Type: [PVBUSlave](#).

`motherboard.virtioblockdevice`

virtio block device.

Type: [VirtioBlockDevice](#).

`motherboard.virtioblockdevice.register_slave`

Type: [PVBUSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.vram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.vram.bus_slave

Type: [PVBUSSlave](#).

periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.20 FVP_VE_Cortex-A9x4

List of instances in FVP_VE_Cortex-A9x4.

FVP_VE_Cortex-A9x4 instances

cluster

ARM CORTEXA9MP Cluster CT model.

Type: [Cluster_ARM_Cortex-A9MP](#).

cluster.acp_mapper

Type: [PVBUSMapper](#).

cluster.cpu0

ARM CORTEXA9MP CT model.

Type: [ARM_Cortex-A9MP](#).

cluster.cpu0.UTLB

Type: [TLB](#).

cluster.cpu0.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster.cpu0.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster.cpu0.utlb

TLB - instruction, data or unified.

Type: [TLBCADI](#).

cluster.cpu1

ARM CORTEXA9MP CT model.

Type: [ARM_Cortex-A9MP](#).

cluster.cpu1.UTLB

Type: [TLB](#).

cluster.cpu1.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu1.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster.cpu1.l1icache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu1.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cluster.cpu1.utlb

TLB - instruction, data or unified.

Type: [TLBCADI](#).

cluster.cpu2

ARM CORTEXA9MP CT model.

Type: [ARM_Cortex-A9MP](#).

cluster.cpu2.UTLB

Type: [TLB](#).

cluster.cpu2.l1dcache

PV Cache.

Type: [PVCACHE](#).

cluster.cpu2.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cluster.cpu2.l1icache

PV Cache.

Type: `PVCache`.**cluster.cpu2.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu2.utlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster.cpu3**

ARM CORTEXA9MP CT model.

Type: `ARM_Cortex-A9MP`.**cluster.cpu3.UTLB**Type: `TLB`.**cluster.cpu3.l1dcache**

PV Cache.

Type: `PVCache`.**cluster.cpu3.l1dcache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu3.l1icache**

PV Cache.

Type: `PVCache`.**cluster.cpu3.l1icache.upstream[0]**Type: `PVBusSlave`.**cluster.cpu3.utlb**

TLB - instruction, data or unified.

Type: `TlbCadi`.**cluster.ext_bus**Type: `PVBusLogger`.**cluster.ext_bus.mapper**Type: `PVBusMapper`.**cluster.internal_shareability_remapper**Type: `PVBusMapper`.**cluster.l1_incoherent_interconnect**Type: `PVCache`.**cluster.l1_incoherent_interconnect.upstream[0]**Type: `PVBusSlave`.**cluster.l1_incoherent_interconnect.upstream[10]**Type: `PVBusSlave`.

cluster.11_incoherent_interconnect.upstream[11]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[12]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[13]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[14]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[15]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cluster.11_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cluster.12_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard

Daughtercard, inspired by the ARM Versatile Express development platform.

Type: [VEDaughterBoard](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.coresight_mapper

Type: [PVBusMapper](#).

daughterboard.dmc

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc.bus_slave

Type: [PVBusSlave](#).

daughterboard.dmc_phy

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dmc_phy.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.dram_aliased

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_aliased.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_4

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_4.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.dram_limit_8

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.dram_limit_8.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.hdlcd

ARM PrimeCell HD Color LCD Controller (Nominal Designation PL370).

Type: [PL370_HDLCD](#).

daughterboard.hdlcd.busmaster

Type: [PVBusMaster](#).

daughterboard.hdlcd.busslave

Type: [PVBusSlave](#).

daughterboard.hdlcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.hdlcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.hdlcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.hdlcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.introuter

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

daughterboard.nonsecure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.nonsecure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.secureDRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureDRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secureRO

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

daughterboard.secureRO.map

Type: [PVBusMapper](#).

daughterboard.secureRO.mbs

Type: [PVBusSlave](#).

daughterboard.secureRO.rmbs

Type: [PVBusSlave](#).

daughterboard.secureROloader

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

daughterboard.secureSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.secureSRAM.bus_slave

Type: [PVBusSlave](#).

daughterboard.secure_region

Allow TrustZone secure/normal bus signals to be routed separately.

Type: [TZSwitch](#).

daughterboard.secure_region.pvbus_mapper

Type: [PVBusMapper](#).

daughterboard.sram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.sram.bus_slave

Type: [PVBusSlave](#).

daughterboard.vedcc

Daughterboard Configuration Control (DCC).

Type: [VEDCC](#).

motherboard

Model inspired by the ARM Versatile Express Motherboard.

Type: [VEMotherBoard](#).

motherboard.Timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_0_1.busslave

Type: [PVBusSlave](#).

motherboard.Timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.Timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

motherboard.Timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

motherboard.Timer_2_3.busslave

Type: [PVBusSlave](#).

motherboard.Timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.clk_div1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.Timer_2_3.counter0`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.Timer_2_3.counter1`

Internal component used by SP804 Timer module.

Type: CounterModule.

`motherboard.audioout`

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

`motherboard.clock100Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock24MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock35MHz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clock50Hz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.clockCLCD`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.dummy_local_dap_rom`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_local_dap_rom.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_ram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_ram.bus_slave`

Type: [PVBusSlave](#).

`motherboard.dummy_usb`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.dummy_usb.bus_slave`

Type: [PVBusSlave](#).

`motherboard.flash0`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash0.map`

Type: [PVBusMapper](#).

`motherboard.flash0.mbs`

Type: [PVBusSlave](#).

`motherboard.flash0.rmbs`

Type: [PVBusSlave](#).

`motherboard.flash1`

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

`motherboard.flash1.map`

Type: [PVBusMapper](#).

`motherboard.flash1.mbs`

Type: [PVBusSlave](#).

`motherboard.flash1.rmbs`

Type: [PVBusSlave](#).

`motherboard.flashloader0`

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

`motherboard.flashloader1`

A device that can preload a gzipped image into flash at startup.

Type: `FlashLoader`.

`motherboard.hostbridge`

Host Socket Interface Component.

Type: `HostBridge`.

`motherboard.mmc`

Generic Multimedia Card.

Type: `MMC`.

`motherboard.mmc.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`motherboard.mmc.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This means that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`motherboard.mmc.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`motherboard.mmc.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`motherboard.pl011_uart0`

ARM PrimeCell UART(PL011).

Type: `PL011_Uart`.

`motherboard.pl011_uart0.busslave`

Type: `PVBusSlave`.

`motherboard.pl011_uart0.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart0.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.pl011_uart0.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart0.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart0.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart1`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart1.busslave`

Type: [PVBUSlave](#).

`motherboard.pl011_uart1.clk_divider`

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart1.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

motherboard.pl011_uart2.busslave

Type: [PVBUSlave](#).

motherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart2.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart2.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl011_uart3`

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

`motherboard.pl011_uart3.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl011_uart3.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl011_uart3.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl011_uart3.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl011_uart3.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl011_uart3.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl031_rtc`

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

`motherboard.pl031_rtc.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl031_rtc.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl031_rtc.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl031_rtc.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl031_rtc.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl041_aaci`

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

`motherboard.pl041_aaci.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl041_aaci.timer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.pl041_aaci.timer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.pl041_aaci.timer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.pl041_aaci.timer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.pl050_kmi0`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi0.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi0.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.pl050_kmi1`

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

`motherboard.pl050_kmi1.busslave`

Type: [PVBUSSlave](#).

`motherboard.pl050_kmi1.clk_divider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

motherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

motherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

motherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

motherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

`motherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

`motherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread64](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.ps2keyboard.ps2_clocktimer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.ps2mouse`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

`motherboard.ps2mouse.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

motherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

motherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

motherboard.psram.bus_slave

Type: [PVBUSSlave](#).

motherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

motherboard.smsc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

motherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

motherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

motherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a

proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.sp805_wdog.clocktimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.sp805_wdog.clocktimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.sp810_sysctrl`

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

`motherboard.sp810_sysctrl.busslave`

Type: [PVBusSlave](#).

`motherboard.sp810_sysctrl.clkdiv_clk0`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk1`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk2`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.sp810_sysctrl.clkdiv_clk3`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

motherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

motherboard.ve_sysregs

Type: [VE_SysRegs](#).

motherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice

virtio block device.

Type: [VirtioBlockDevice](#).

motherboard.virtioblockdevice.register_slave

Type: [PVBusSlave](#).

motherboard.virtioblockdevice.virtio_master

Type: [PVBusMaster](#).

motherboard.virtiop9device

virtio P9 server.

Type: [VirtioP9Device](#).

motherboard.virtiop9device.mmio_slave

Type: [PVBusSlave](#).

motherboard.virtiop9device.virtio_master

Type: [PVBusMaster](#).

motherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

motherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

motherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vis.recorder.playbackTimer`

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

`motherboard.vis.recorder.playbackTimer.timer`

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

`motherboard.vis.recorder.playbackTimer.timer.thread`

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

`motherboard.vis.recorder.playbackTimer.timer.thread_event`

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

`motherboard.vis.recorder.recordingDivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`motherboard.vram`

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

`motherboard.vram.bus_slave`

Type: [PVBusSlave](#).

`periph_clockdivider`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.21 FVP_VE_Cortex-R4

List of instances in FVP_VE_Cortex-R4.

FVP_VE_Cortex-R4 instances

daughterboard

Cortex-R4 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R4](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.core

ARM CORTEXR4 CT model.

Type: [ARM_Cortex-R4](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.ext_bus

Type: [PVBusLogger](#).

daughterboard.core.ext_bus.mapper

Type: [PVBusMapper](#).

daughterboard.core.l1_incoherent_interconnect

Type: [PVCache](#).

daughterboard.core.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBusSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBusSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBusMapper](#).

daughterboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

daughterboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

daughterboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

daughterboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

daughterboard.pl111_clcd.pl11x_clcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.pl390_gic

Generic Interrupt Controller (PL390).

Type: [PL390_GIC](#).

daughterboard.pl390_gic.busslave_cpu

Type: [PVBusSlave](#).

daughterboard.pl390_gic.busslave_distributor

Type: [PVBusSlave](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

vemotherboard.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

vemotherboard.pl180_mci.busslave

Type: `PVBusSlave`.

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: `PS2Keyboard`.

vemotherboard.ps2keyboard.ps2_clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.22 FVP_VE_Cortex-R5x1

List of instances in FVP_VE_Cortex-R5x1.

FVP_VE_Cortex-R5x1 instances

daughterboard

Cortex-R5x1 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R5x1](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

daughterboard.core

ARM CORTEXR5 Cluster CT model.

Type: `Cluster_ARM_Cortex-R5`.

daughterboard.core.acp_mapper

Type: `PVBusMapper`.

daughterboard.core.cpu0

ARM CORTEXR5 CT model.

Type: `ARM_Cortex-R5`.

daughterboard.core.cpu0.l1dcache

Type: `PVCache`.

daughterboard.core.cpu0.l1dcache.upstream[0]

Type: `PVBusSlave`.

daughterboard.core.cpu0.l1icache

Type: `PVCache`.

daughterboard.core.cpu0.l1icache.upstream[0]

Type: `PVBusSlave`.

daughterboard.core.ext_bus

Type: `PVBusLogger`.

daughterboard.core.ext_bus.mapper

Type: `PVBusMapper`.

daughterboard.core.l1_incoherent_interconnect

Type: `PVCache`.

daughterboard.core.l1_incoherent_interconnect.upstream[0]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[10]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[11]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[12]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[13]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[14]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[15]

Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

daughterboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

daughterboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

daughterboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

daughterboard.pl111_clcd.pl11x_clcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.pl390_gic

Generic Interrupt Controller (PL390).

Type: [PL390_GIC](#).

daughterboard.pl390_gic.busslave_cpu

Type: [PVBUSSlave](#).

daughterboard.pl390_gic.busslave_distributor

Type: [PVBUSSlave](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

vemotherboard.pl180_mci.busslave

Type: [PVBUSSlave](#).

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

vemotherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBUSSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBUSSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.23 FVP_VE_Cortex-R5x2

List of instances in FVP_VE_Cortex-R5x2.

FVP_VE_Cortex-R5x2 instances

daughterboard

Cortex-R5x2 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R5x2](#).

daughterboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.core

ARM CORTEXR5 Cluster CT model.

Type: [Cluster_ARM_Cortex-R5](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0

ARM CORTEXR5 CT model.

Type: [ARM_Cortex-R5](#).

daughterboard.core.cpu0.l1dcache

Type: [PVCache](#).

daughterboard.core.cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

daughterboard.core.cpu0.l1icache

Type: [PVCache](#).

daughterboard.core.cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

daughterboard.core.cpu1

ARM CORTEXR5 CT model.

Type: [ARM_Cortex-R5](#).

daughterboard.core.cpu1.l1dcache

Type: [PVCache](#).

daughterboard.core.cpu1.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

daughterboard.core.cpu1.l1icache

Type: [PVCache](#).

daughterboard.core.cpu1.l1icache.upstream[0]

Type: [PVBUSSlave](#).

daughterboard.core.ext_bus

Type: [PVBUSLogger](#).

daughterboard.core.ext_bus.mapper

Type: [PVBUSMapper](#).

daughterboard.core.l1_incoherent_interconnect

Type: [PVCache](#).

daughterboard.core.l1_incoherent_interconnect.upstream[0]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[10]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[11]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[12]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[13]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[14]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[15]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

daughterboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

daughterboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

daughterboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

daughterboard.pl111_clcd.pl11x_clcd.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

daughterboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.pl390_gic

Generic Interrupt Controller (PL390).

Type: [PL390_GIC](#).

daughterboard.pl390_gic.busslave_cpu

Type: [PVBUSSlave](#).

daughterboard.pl390_gic.busslave_distributor

Type: [PVBUSSlave](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

vemotherboard.pl180_mci.busslave

Type: [PVBUSSlave](#).

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

vemotherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.24 FVP_VE_Cortex-R7x1

List of instances in FVP_VE_Cortex-R7x1.

FVP_VE_Cortex-R7x1 instances

clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard

Cortex_R7x1 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R7x1](#).

daughterboard.core

ARM CORTEXR7 Cluster CT model.

Type: [Cluster_ARM_Cortex-R7](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0

ARM CORTEXR7 CT model.

Type: [ARM_Cortex-R7](#).

daughterboard.core.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

`daughterboard.core.cpu0.l1icache.upstream[0]`
Type: `PVBusSlave`.

`daughterboard.core.ext_bus`
Type: `PVBusLogger`.

`daughterboard.core.ext_bus.mapper`
Type: `PVBusMapper`.

`daughterboard.core.l1_incoherent_interconnect`
Type: `PVCache`.

`daughterboard.core.l1_incoherent_interconnect.upstream[0]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[10]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[11]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[12]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[13]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[14]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[15]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[16]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[17]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[1]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[2]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[3]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[4]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[5]`
Type: `PVBusSlave`.

`daughterboard.core.l1_incoherent_interconnect.upstream[6]`
Type: `PVBusSlave`.

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for

the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke

wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBusSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBusSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

vemotherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

vemotherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBUSSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBUSSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBUSSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.25 FVP_VE_Cortex-R7x2

List of instances in FVP_VE_Cortex-R7x2.

FVP_VE_Cortex-R7x2 instances

clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard

Cortex_R7x2 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R7x2](#).

daughterboard.core

ARM CORTEXR7 Cluster CT model.

Type: [Cluster_ARM_Cortex-R7](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0

ARM CORTEXR7 CT model.

Type: [ARM_Cortex-R7](#).

daughterboard.core.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu1

ARM CORTEXR7 CT model.

Type: [ARM_Cortex-R7](#).

daughterboard.core.cpu1.l1dcache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu1.l1icache
PV Cache.
Type: [PVCache](#).

daughterboard.core.cpu1.l1icache.upstream[0]
Type: [PVBusSlave](#).

daughterboard.core.ext_bus
Type: [PVBusLogger](#).

daughterboard.core.ext_bus.mapper
Type: [PVBusMapper](#).

daughterboard.core.l1_incoherent_interconnect
Type: [PVCache](#).

daughterboard.core.l1_incoherent_interconnect.upstream[0]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[10]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[11]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[12]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[13]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[14]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[15]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[16]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]
Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component

which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl041_aaci.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBusSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBusMaster](#).

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBusSlave](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

vemotherboard.pl180_mci.busslave

Type: [PVBusSlave](#).

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

vemotherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.26 FVP_VE_Cortex-R8x1

List of instances in FVP_VE_Cortex-R8x1.

FVP_VE_Cortex-R8x1 instances

clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard

Cortex_R8x1 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R8x1](#).

daughterboard.core

ARM CORTEXR8 Cluster CT model.

Type: [Cluster_ARM_Cortex-R8](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).

daughterboard.core.cpu0.l1dcache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu0.l1icache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.ext_bus

Type: [PVBusLogger](#).

daughterboard.core.ext_bus.mapper

Type: [PVBusMapper](#).

daughterboard.core.l1_incoherent_interconnect

Type: [PVCache](#).

daughterboard.core.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[10]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[11]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[12]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[13]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[14]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[15]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBusSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBusSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: `PVBusMaster`.

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: `PVBusSlave`.

vemotherboard.pl111_clcd.pl11x_clcd.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

vemotherboard.pl180_mci.busslave

Type: `PVBusSlave`.

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: `PS2Keyboard`.

vemotherboard.ps2keyboard.ps2_clocktimer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.27 FVP_VE_Cortex-R8x2

List of instances in FVP_VE_Cortex-R8x2.

FVP_VE_Cortex-R8x2 instances

clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard

Cortex_R8x2 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R8x2](#).

daughterboard.core

ARM CORTEXR8 Cluster CT model.

Type: [Cluster_ARM_Cortex-R8](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).**daughterboard.core.cpu0.l1dcache**

PV Cache.

Type: [PVCache](#).**daughterboard.core.cpu0.l1dcache.upstream[0]**Type: [PVBusSlave](#).**daughterboard.core.cpu0.l1icache**

PV Cache.

Type: [PVCache](#).**daughterboard.core.cpu0.l1icache.upstream[0]**Type: [PVBusSlave](#).**daughterboard.core.cpu1**

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).**daughterboard.core.cpu1.l1dcache**

PV Cache.

Type: [PVCache](#).**daughterboard.core.cpu1.l1dcache.upstream[0]**Type: [PVBusSlave](#).**daughterboard.core.cpu1.l1icache**

PV Cache.

Type: [PVCache](#).**daughterboard.core.cpu1.l1icache.upstream[0]**Type: [PVBusSlave](#).**daughterboard.core.ext_bus**Type: [PVBusLogger](#).**daughterboard.core.ext_bus.mapper**Type: [PVBusMapper](#).**daughterboard.core.l1_incoherent_interconnect**Type: [PVCache](#).**daughterboard.core.l1_incoherent_interconnect.upstream[0]**Type: [PVBusSlave](#).**daughterboard.core.l1_incoherent_interconnect.upstream[10]**Type: [PVBusSlave](#).**daughterboard.core.l1_incoherent_interconnect.upstream[11]**Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[12]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[13]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[14]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[15]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[16]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.periph_clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBusSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBusSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: `PL11x_CLCD`.

`vemotherboard.pl111_clcd.pl11x_clcd.busmaster`

Type: `PVBusMaster`.

`vemotherboard.pl111_clcd.pl11x_clcd.busslave`

Type: `PVBusSlave`.

`vemotherboard.pl111_clcd.pl11x_clcd.timer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

`vemotherboard.pl111_clcd.pl11x_clcd.timer.timer`

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

`vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread`

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

`vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event`

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

`vemotherboard.pl180_mci`

ARM PrimeCell Multimedia Card Interface (PL180).

Type: `PL180_MCI`.

`vemotherboard.pl180_mci.busslave`

Type: `PVBusSlave`.

`vemotherboard.ps2keyboard`

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked `PS2Data` signals which can be routed to a `PL050_KMI` component.

Type: `PS2Keyboard`.

`vemotherboard.ps2keyboard.ps2_clocktimer`

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()`

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBusSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

14.28 FVP_VE_Cortex-R8x4

List of instances in FVP_VE_Cortex-R8x4.

FVP_VE_Cortex-R8x4 instances

clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard

Cortex_R8x4 DaughterBoard for Versatile Express.

Type: [VEDaughterBoardCortex_R8x4](#).

daughterboard.core

ARM CORTEXR8 Cluster CT model.

Type: [Cluster_ARM_Cortex-R8](#).

daughterboard.core.acp_mapper

Type: [PVBusMapper](#).

daughterboard.core.cpu0

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).**daughterboard.core.cpu0.l1dcache**

PV Cache.

Type: [pVCache](#).**daughterboard.core.cpu0.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**daughterboard.core.cpu0.l1icache**

PV Cache.

Type: [pVCache](#).**daughterboard.core.cpu0.l1icache.upstream[0]**Type: [PVBUSSlave](#).**daughterboard.core.cpu1**

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).**daughterboard.core.cpu1.l1dcache**

PV Cache.

Type: [pVCache](#).**daughterboard.core.cpu1.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**daughterboard.core.cpu1.l1icache**

PV Cache.

Type: [pVCache](#).**daughterboard.core.cpu1.l1icache.upstream[0]**Type: [PVBUSSlave](#).**daughterboard.core.cpu2**

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).**daughterboard.core.cpu2.l1dcache**

PV Cache.

Type: [pVCache](#).**daughterboard.core.cpu2.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**daughterboard.core.cpu2.l1icache**

PV Cache.

Type: [pVCache](#).**daughterboard.core.cpu2.l1icache.upstream[0]**Type: [PVBUSSlave](#).

daughterboard.core.cpu3

ARM CORTEXR8 CT model.

Type: [ARM_Cortex-R8](#).

daughterboard.core.cpu3.l1dcache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu3.l1dcache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.cpu3.l1icache

PV Cache.

Type: [PVCache](#).

daughterboard.core.cpu3.l1icache.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.ext_bus

Type: [PVBusLogger](#).

daughterboard.core.ext_bus.mapper

Type: [PVBusMapper](#).

daughterboard.core.l1_incoherent_interconnect

Type: [PVCache](#).

daughterboard.core.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[1]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

daughterboard.core.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

daughterboard.core.l2_flusher

Type: [AsyncCacheFlushUnit](#).

daughterboard.dram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

daughterboard.dram.bus_slave

Type: [PVBUSSlave](#).

daughterboard.exclusive_monitor

Global exclusive monitor.

Type: [PVBUSExclusiveMonitor](#).

daughterboard.exclusive_monitor.bus_mapper

Type: [PVBUSMapper](#).

daughterboard.periph_clockdivider

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

daughterboard.pl310_l2cc

ARM PrimeCell Level 2 Cache Controller (PL310).

Type: [PL310_L2CC](#).

daughterboard.veinterruptmapper

Interrupt Mapping peripheral (non-cascaded).

Type: [VEInterruptMapper](#).

vemotherboard

Model inspired by the ARM Versatile Express Motherboard for R profile.

Type: [VEMotherBoardR](#).

vemotherboard.audioout

SDL based Audio Output for PL041_AACI.

Type: [AudioOut_SDL](#).

vemotherboard.clock100Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock24MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock35MHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.clockCLCD

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.dummy_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.dummy_usb

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.dummy_usb.bus_slave

Type: [PVBusSlave](#).

vemotherboard.flash0

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash0.map

Type: [PVBusMapper](#).

vemotherboard.flash0.mbs

Type: [PVBusSlave](#).

vemotherboard.flash0.rmbs

Type: [PVBusSlave](#).

vemotherboard.flash1

Intel Strata Flash J3 LISA+ model.

Type: [IntelStrataFlashJ3](#).

vemotherboard.flash1.map

Type: [PVBusMapper](#).

vemotherboard.flash1.mbs

Type: [PVBusSlave](#).

vemotherboard.flash1.rmbs

Type: [PVBusSlave](#).

vemotherboard.flashloader0

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.flashloader1

A device that can preload a gzipped image into flash at startup.

Type: [FlashLoader](#).

vemotherboard.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

vemotherboard.mmc

Generic Multimedia Card.

Type: [MMC](#).

vemotherboard.mmc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.mmc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.mmc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.mmc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart0

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart0.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart1

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl011_uart1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart2

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart2.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart3

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart3.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart3.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl011_uart4

ARM PrimeCell UART(PL011).

Type: [PL011_Uart](#).

vemotherboard.pl011_uart4.busslave

Type: [PVBusSlave](#).

vemotherboard.pl011_uart4.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl011_uart4.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl011_uart4.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl011_uart4.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl011_uart4.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl031_rtc

ARM PrimeCell Real Time Clock(PL031).

Type: [PL031_RTC](#).

vemotherboard.pl031_rtc.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl031_rtc.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl031_rtc.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl031_rtc.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl031_rtc.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl041_aaci

ARM PrimeCell Advanced Audio CODEC Interface(PL041).

Type: [PL041_AACI](#).

vemotherboard.pl041_aaci.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl041_aaci.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl041_aaci.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl041_aaci.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl041_aaci.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl050_kmi0

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi0.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl050_kmi1

ARM PrimeCell PS2 Keyboard/Mouse Interface(PL050).

Type: [PL050_KMI](#).

vemotherboard.pl050_kmi1.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl050_kmi1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.pl111_clcd

ARM PrimeCell Color LCD Controller(PL111).

Type: [PL111_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd

Internal component used by PL110 and PL111 CLCD controllers.

Type: [PL11x_CLCD](#).

vemotherboard.pl111_clcd.pl11x_clcd.busmaster

Type: [PVBUSMaster](#).

vemotherboard.pl111_clcd.pl11x_clcd.busslave

Type: [PVBUSSlave](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.pl111_clcd.pl11x_clcd.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.pl180_mci

ARM PrimeCell Multimedia Card Interface (PL180).

Type: [PL180_MCI](#).

vemotherboard.pl180_mci.busslave

Type: [PVBUSSlave](#).

vemotherboard.ps2keyboard

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Keyboard](#).

vemotherboard.ps2keyboard.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2keyboard.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.ps2mouse

Interface component, which takes the keypress/release signals from the Visualisation component and translates them into clocked PS2Data signals which can be routed to a PL050_KMI component.

Type: [PS2Mouse](#).

vemotherboard.ps2mouse.ps2_clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.ps2mouse.ps2_clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.psram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.psram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.smisc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

vemotherboard.smc_91c111.SMSC_slave

Type: [PVBUSSlave](#).

vemotherboard.sp805_wdog

ARM Watchdog Module(SP805).

Type: [SP805_Watchdog](#).

vemotherboard.sp805_wdog.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp805_wdog.clocktimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.sp805_wdog.clocktimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.sp810_sysctrl

PrimeXsys System Controller(SP810) NB: Only EB relevant functionalities are fully implemented.

Type: [SP810_SysCtrl](#).

vemotherboard.sp810_sysctrl.busslave

Type: [PVBUSSlave](#).

vemotherboard.sp810_sysctrl.clkdiv_clk0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk2

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.sp810_sysctrl.clkdiv_clk3

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.terminal_0

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_1

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_2

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_3

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.terminal_4

Telnet terminal interface.

Type: [TelnetTerminal](#).

vemotherboard.timer_0_1

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_0_1.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_0_1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_0_1.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_0_1.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3

ARM Dual-Timer Module(SP804).

Type: [SP804_Timer](#).

vemotherboard.timer_2_3.busslave

Type: [PVBusSlave](#).

vemotherboard.timer_2_3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.timer_2_3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.timer_2_3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

vemotherboard.ve_sysregs

Type: [VE_SysRegs](#).

vemotherboard.ve_sysregs.busslave

Type: [PVBusSlave](#).

vemotherboard.video_ram

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

vemotherboard.video_ram.bus_slave

Type: [PVBusSlave](#).

vemotherboard.vis

Display window for VE using Visualisation library.

Type: [VEVisualisation](#).

vemotherboard.vis.recorder

Event recorder component for visualisation component (allows to playback and record interactive GUI sessions).

Type: [VisEventRecorder](#).

vemotherboard.vis.recorder.playbackDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

vemotherboard.vis.recorder.playbackTimer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

vemotherboard.vis.recorder.playbackTimer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

vemotherboard.vis.recorder.recordingDivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

15. MPS2 Platform FVPs

This chapter describes the MPS2 Platform FVPs contained in the FVP Standard Library.

For the MPS2 memory maps, see [MPS2 - memory maps](#) in the *Fast Models Reference Guide*.

- The MPS2 FVPs include AVH peripherals. These peripherals require the standard Python libraries, which are installed in the `<FVP_INSTALL_PATH>/python/lib/` directory. To set up the environment variables for these FVPs, including automatically configuring them to use the standard Python libraries, run the command:



Note

```
source $FVP_INSTALL_PATH/scripts/runtime.sh
```

Alternatively, manually set the `PYTHONHOME` environment variable to the location of the standard Python libraries.

- When running a project built using GCC and CMSIS on a Cortex®-M FVP with semihosting enabled, the semihosting implementation can overwrite the memory configuration.

To avoid this problem, configure the required memory values using model parameters, as described in this Knowledge Base Article [How do I Avoid Stack Pointer Corruption When Semihosting is Enabled on a GCC Toolchain?](#).

15.1 FVP_MPS2_AEMv8M

List of instances in FVP_MPS2_AEMv8M.

About FVP_MPS2_AEMv8M

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \  
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_AEMv8M instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cpu0

ARM AEMv8M CT model.

Type: [ARM_AEMv8M](#).

cpu0.acp_mapper

Type: [PVBusMapper](#).

cpu0.ext_bus

Type: [PVBusLogger](#).

cpu0.ext_bus.mapper

Type: [PVBusMapper](#).

cpu0.l1_incoherent_interconnect

Type: [PVCache](#).

cpu0.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[2]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[3]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[4]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[5]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[6]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[7]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[8]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[9]

Type: [PVBUSSlave](#).

cpu0.l1dcache

Type: [PVCACHE](#).

cpu0.l1dcache.upstream[0]

Type: [PVBUSSlave](#).

cpu0.l1icache

Type: [PVCACHE](#).

cpu0.l1icache.upstream[0]

Type: [PVBUSSlave](#).

cpu0.l2_flusher

Type: [ASYNCCACHEFLUSHUNIT](#).

cpu1

ARM AEMv8M CT model.

Type: [ARM_AEMv8M](#).

cpu1.acp_mapper

Type: [PVBUSMAPPER](#).

cpu1.ext_bus

Type: [PVBUSLOGGER](#).

cpu1.ext_bus.mapper

Type: [PVBUSMAPPER](#).

cpu1.l1_incoherent_interconnect

Type: [PVCACHE](#).

cpu1.l1_incoherent_interconnect.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[10]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[11]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[12]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[13]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[14]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[15]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[16]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[17]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[1]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[2]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu1.l1dcache
Type: [PVCACHE](#).

cpu1.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cpu1.l1icache

Type: [PVCache](#).

cpu1.l1icache.upstream[0]

Type: [PVBusSlave](#).

cpu1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

fvp_mps2

MPS2 DUT.

Type: [FVP_MPS2](#).

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO0.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO1.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO2.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO3.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test

Type: [GPIO_Connection_Test](#).

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: [GPIO_Port_Transfer](#).

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: [GPIO1_Connection_Test](#).

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbusslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbusslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion2
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMACE](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_securitymodifier
Type: SecurityModifier.

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_squasher
Squashes the exclusive attribute on bus transactions.
Type: PVBusExclusiveSquasher.

fvp_mps2.exclusive_squasher.bus_modifier
Type: PVBusMapper.

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbuslave

Type: [PVBUSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer1.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBUSSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBUSMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: PVBUSMapper.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_switch_dma1.mapper

Type: PVBUSMapper.

fvp_mps2.signal_router

Signal router.

Type: SignalRouter.

fvp_mps2.smc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC slave
Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate
Or Gate.
Type: [OrGate](#).

fvp_mps2.spniden_or_gate
Or Gate.
Type: [OrGate](#).

fvp_mps2.sse200
SSE-200 subsystem.
Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.crypto_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slaveType: [PVBusSlave](#).**fvp_mps2.stub_spi2**

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).**fvp_mps2.stub_spi2.bus_slave**Type: [PVBusSlave](#).**fvp_mps2.svos_dualtimer**Type: [SVOS_DualTimer](#).**fvp_mps2.svos_dualtimer.busmaster0**Type: [PVBusMaster](#).**fvp_mps2.svos_dualtimer.busmaster1**Type: [PVBusMaster](#).**fvp_mps2.svos_dualtimer.busmaster2**Type: [PVBusMaster](#).**fvp_mps2.svos_dualtimer.busmaster3**Type: [PVBusMaster](#).**fvp_mps2.svos_dualtimer.busslave**Type: [PVBusSlave](#).**fvp_mps2.svos_dualtimer.svos_dualtimer0**

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).**fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave**Type: [PVBusSlave](#).**fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal1**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal2**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.touchscreen_interface**

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).**fvp_mps2.touchscreen_interface.pvbuslave**Type: [PVBusSlave](#).**fvp_mps2.uart_overflows_or_gate**

Or Gate.

Type: [OrGate](#).**idau.bus_bridge**

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.2 FVP_MPS2_Cortex-M0

List of instances in FVP_MPS2_Cortex-M0.

FVP_MPS2_Cortex-M0 instances

armcortexm0ct

ARM CORTEXM0 CT model.

Type: [ARM_Cortex-M0](#).**armcortexm0ct.acp_mapper**Type: [PVBusMapper](#).**armcortexm0ct.ext_bus**Type: [PVBusLogger](#).**armcortexm0ct.ext_bus.mapper**Type: [PVBusMapper](#).**armcortexm0ct.l2_flusher**Type: [AsyncCacheFlushUnit](#).

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2

MPS2 DUT.

Type: FVP_MPS2.

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO_connection_test

Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: `GPIO1_Connection_Test`.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM.bus_slave

Type: `PVBusSlave`.

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM_M7.bus_slave

Type: `PVBusSlave`.

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART0.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.UART0.pvbusslave

Type: `PVBusSlave`.

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART1.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.UART1.pvbusslave

Type: `PVBusSlave`.

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART2.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface.pvbusslave1

Type: [PVBusSlave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifierType: [PVBusMapper](#).**fvp_mps2.dma2**

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).**fvp_mps2.dma2.busmaster0**Type: [PVBusMaster](#).**fvp_mps2.dma2.busmaster1**Type: [PVBusMaster](#).**fvp_mps2.dma2.busslave**Type: [PVBusSlave](#).**fvp_mps2.dma2.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**fvp_mps2.dma2.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**fvp_mps2.dma2.timer.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**fvp_mps2.dma2.timer.timer.thread_event**

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**fvp_mps2.dma2_idau_labeller**Type: [LabellerIdauSecurity](#).**fvp_mps2.dma2_idau_labeller.idau_busmaster**Type: [PVBusMaster](#).**fvp_mps2.dma2_idau_labeller.pvbusmodifier**Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmasterType: [PVBusMaster](#).**fvp_mps2.dma2_securitymodifier**Type: [SecurityModifier](#).**fvp_mps2.dma2_securitymodifier.pvbusmodifier**Type: [PVBusMapper](#).**fvp_mps2.dma3**

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).**fvp_mps2.dma3.busmaster0**Type: [PVBusMaster](#).**fvp_mps2.dma3.busmaster1**Type: [PVBusMaster](#).**fvp_mps2.dma3.busslave**Type: [PVBusSlave](#).**fvp_mps2.dma3.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**fvp_mps2.dma3.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**fvp_mps2.dma3.timer.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**fvp_mps2.dma3.timer.timer.thread_event**

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**fvp_mps2.dma3_idau_labeller**Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster
Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier
Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster
Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier
Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: PVBusSlave.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: PVBusExclusiveMonitor.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: PVBusExclusiveMonitor.

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: IoTSS_MemoryProtectionController.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: PVBusSlave.

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: [PVBusSlave](#).

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_bus_switch.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cordio_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu0core_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu0core_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu0dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu0dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu1core_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu1core_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu1dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu1dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.crypto_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.crypto_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: PVBusSlave.

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram1.bus_slave

Type: PVBusSlave.

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram2.bus_slave

Type: PVBusSlave.

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub0.bus_slave

Type: PVBusSlave.

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub1.bus_slave

Type: PVBusSlave.

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub_i2c1.bus_slave

Type: PVBusSlave.

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).**fvp_mps2.switch_PSRAM_M7.mapper**Type: [PVBusMapper](#).**fvp_mps2.switch_svos_dualtimer**

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).**fvp_mps2.switch_svos_dualtimer.mapper**Type: [PVBusMapper](#).**fvp_mps2.telnetterminal0**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal1**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal2**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.touchscreen_interface**

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).**fvp_mps2.touchscreen_interface.pvbusslave**Type: [PVBusSlave](#).**fvp_mps2.uart_overflows_or_gate**

Or Gate.

Type: [OrGate](#).

15.3 FVP_MPS2_Cortex-M0plus

List of instances in FVP_MPS2_Cortex-M0plus.

FVP_MPS2_Cortex-M0plus instances

armcortexm0plusct

ARM CORTEXM0+ CT model.

Type: [ARM_Cortex-M0+](#).**armcortexm0plusct.acp_mapper**Type: [PVBusMapper](#).

armcortexm0plusct.ext_bus

Type: [PVBusLogger](#).

armcortexm0plusct.ext_bus.mapper

Type: [PVBusMapper](#).

armcortexm0plusct.l2_flusher

Type: [AsyncCacheFlushUnit](#).

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2

MPS2 DUT.

Type: [FVP_MPS2](#).

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO0.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO1.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO2.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO3.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test

Type: [GPIO_Connection_Test](#).

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: [GPIO_Port_Transfer](#).

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: [GPIO1_Connection_Test](#).

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.UART2.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbuslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMACH](#).

fvp_mps2.dma0.busmaster0

Type: [PVBUSMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBUSMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: `MPS2_Audio`.

fvp_mps2.mps2_audio.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: `CMSDK_DualTimer`.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: `MPS2_LCD`.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.mps2_timer1.busslave

Type: PVBusSlave.

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBusSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cordio_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.crypto_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: `IoTSS_CPUIdentity`.

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: `IoTSS_SystemControl`.

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: `PVBusMaster`.

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: `IoTSS_SystemInfo`.

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslaveType: [PVBUSSlave](#).**fvp_mps2.sse200.ram0_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.ram0_ppu.busslave**Type: [PVBUSSlave](#).**fvp_mps2.sse200.ram1_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.ram1_ppu.busslave**Type: [PVBUSSlave](#).**fvp_mps2.sse200.ram2_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.ram2_ppu.busslave**Type: [PVBUSSlave](#).**fvp_mps2.sse200.ram3_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.ram3_ppu.busslave**Type: [PVBUSSlave](#).**fvp_mps2.sse200.s32k_timer**

ARM Timer Module.

Type: [CMSDK_Timer](#).**fvp_mps2.sse200.s32k_timer.busslave**Type: [PVBUSSlave](#).**fvp_mps2.sse200.s32k_timer.clk_div**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.sse200.s32k_timer.counter**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**fvp_mps2.sse200.s32k_watchdog**

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslaveType: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block**

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).**fvp_mps2.sse200.secure_control_register_block.bus_mapper**Type: [PVBusMapper](#).**fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block.busslave_s_iot**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block.idau_busmaster**Type: [PVBusMaster](#).**fvp_mps2.sse200.secure_watchdog**

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).**fvp_mps2.sse200.secure_watchdog.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.signal_router**

Signal router.

Type: [SignalRouter](#).**fvp_mps2.sse200.sys_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.sys_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.timer0**

ARM Timer Module.

Type: [CMSDK_Timer](#).**fvp_mps2.sse200.timer0.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.timer0.clk_div**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2c1
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_i2s
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_spi0
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_spi2
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer
Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0
ARM Dual-Timer Module.
Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.switch_PSRAM_M7.mapper

Type: PVBusMapper.

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.switch_svos_dualtimer.mapper

Type: PVBusMapper.

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: TelnetTerminal.

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: TelnetTerminal.

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: TelnetTerminal.

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: MPS2_TouchScreen.

fvp_mps2.touchscreen_interface.pvbusslave

Type: PVBusSlave.

fvp_mps2.uart_overflows_or_gate

Or Gate.

Type: OrGate.

15.4 FVP_MPS2_Cortex-M23

List of instances in FVP_MPS2_Cortex-M23.

About FVP_MPS2_Cortex-M23

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of

these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_Cortex-M23 instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

core0_bus_gasket

STLBusGasket allows a debugger or emulated T32 code to force the results of system-register reads by writing an address to the ADDR register then 32-bit values to the VALUE register, which are placed in a fifo associated with that address. A PVBUS transaction into pvbus_in goes unchanged to pvbus_out, unless its address matches that associated with a non-empty fifo, in which case: writes are ignored, non-word reads abort, and word reads take values from that fifo.

Type: [STLBusGasket](#).

core0_bus_gasket.busmapper

Type: [PVBUSMapper](#).

cpu0

ARM CORTEXM23 CT model.

Type: [ARM_Cortex-M23](#).

cpu0.acp_mapper

Type: [PVBUSMapper](#).

cpu0.ext_bus

Type: [PVBUSLogger](#).

cpu0.ext_bus.mapper

Type: [PVBUSMapper](#).

cpu0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cpu1

ARM CORTEXM23 CT model.

Type: [ARM_Cortex-M23](#).

cpu1.acp_mapper
Type: [PVBusMapper](#).

cpu1.ext_bus
Type: [PVBusLogger](#).

cpu1.ext_bus.mapper
Type: [PVBusMapper](#).

cpu1.l2_flusher
Type: [AsyncCacheFlushUnit](#).

fvp_mps2
MPS2 DUT.
Type: [FVP_MPS2](#).

fvp_mps2.GPIO0
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO0.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO1
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO1.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO2
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO2.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO3
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO3.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test
Type: [GPIO_Connection_Test](#).

fvp_mps2.GPIO_connection_test.GPIO0_port_trans
Type: [GPIO_Port_Transfer](#).

fvp_mps2.GPIO_connection_test.GPIO1_port_test
Type: [GPIO1_Connection_Test](#).

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface.pvbusslave1

Type: [PVBusSlave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: `IoTSS_PeripheralProtectionController`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8**Type: `PVBusMapper`.**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9**Type: `PVBusMapper`.**fvp_mps2.clock50Hz**

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMALC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly

or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMACE](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_securitymodifier
Type: SecurityModifier.

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_squasher
Squashes the exclusive attribute on bus transactions.
Type: PVBusExclusiveSquasher.

fvp_mps2.exclusive_squasher.bus_modifier
Type: PVBusMapper.

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbuslave

Type: [PVBUSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: IoTSS_MemoryProtectionController.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer1.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBusSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma1.mapper

Type: PVBusMapper.

fvp_mps2.signal_router

Signal router.

Type: SignalRouter.

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.crypto_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [SVOS_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal1**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal2**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.touchscreen_interface**

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).**fvp_mps2.touchscreen_interface.pvbuslave**Type: [PVBusSlave](#).**fvp_mps2.uart_overflows_or_gate**

Or Gate.

Type: [OrGate](#).**idau.bus_bridge**

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.5 FVP_MPS2_Cortex-M3

List of instances in FVP_MPS2_Cortex-M3.

FVP_MPS2_Cortex-M3 instances

armcortexm3ct

ARM CORTEXM3 CT model.

Type: [ARM_Cortex-M3](#).**armcortexm3ct.acp_mapper**Type: [PVBusMapper](#).**armcortexm3ct.ext_bus**Type: [PVBusLogger](#).**armcortexm3ct.ext_bus.mapper**Type: [PVBusMapper](#).**armcortexm3ct.l2_flusher**Type: [AsyncCacheFlushUnit](#).

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2

MPS2 DUT.

Type: FVP_MPS2.

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.GPIO_connection_test

Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: `GPIO1_Connection_Test`.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM.bus_slave

Type: `PVBusSlave`.

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM_M7.bus_slave

Type: `PVBusSlave`.

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART0.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.UART0.pvbusslave

Type: `PVBusSlave`.

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART1.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.UART1.pvbusslave

Type: `PVBusSlave`.

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART2.clk_divider

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface.pvbusslave1

Type: [PVBusSlave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmasterType: [PVBusMaster](#).**fvp_mps2.dma2_securitymodifier**Type: [SecurityModifier](#).**fvp_mps2.dma2_securitymodifier.pvbusmodifier**Type: [PVBusMapper](#).**fvp_mps2.dma3**

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).**fvp_mps2.dma3.busmaster0**Type: [PVBusMaster](#).**fvp_mps2.dma3.busmaster1**Type: [PVBusMaster](#).**fvp_mps2.dma3.busslave**Type: [PVBusSlave](#).**fvp_mps2.dma3.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**fvp_mps2.dma3.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**fvp_mps2.dma3.timer.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**fvp_mps2.dma3.timer.timer.thread_event**A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.Type: [SchedulerThreadEvent](#).**fvp_mps2.dma3_idau_labeller**Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster
Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier
Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster
Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier
Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: PVBusSlave.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: PVBusExclusiveMonitor.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: PVBusExclusiveMonitor.

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: IoTSS_MemoryProtectionController.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: PVBusSlave.

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: [PVBusSlave](#).

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_bus_switch.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cordio_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu0core_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu0core_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu0dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu0dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu1core_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu1core_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu1dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu1dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.crypto_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.crypto_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: PVBusSlave.

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram1.bus_slave

Type: PVBusSlave.

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram2.bus_slave

Type: PVBusSlave.

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub0.bus_slave

Type: PVBusSlave.

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub1.bus_slave

Type: PVBusSlave.

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub_i2c1.bus_slave

Type: PVBusSlave.

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).

fvp_mps2.touchscreen_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.uart_overflows_or_gate

Or Gate.

Type: [OrGate](#).

15.6 FVP_MPS2_Cortex-M33

List of instances in FVP_MPS2_Cortex-M33.

About FVP_MPS2_Cortex-M33

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_Cortex-M33 instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cpu0

ARM CORTEXM33 CT model.

Type: [ARM_Cortex-M33](#).

cpu0.acp_mapper

Type: [PVBusMapper](#).

cpu0.ext_bus

Type: [PVBusLogger](#).

cpu0.ext_bus.mapper

Type: [PVBusMapper](#).

cpu0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cpu1

ARM CORTEXM33 CT model.

Type: [ARM_Cortex-M33](#).

cpu1.acp_mapper

Type: [PVBusMapper](#).

cpu1.ext_bus

Type: [PVBusLogger](#).

cpu1.ext_bus.mapper

Type: [PVBusMapper](#).

cpu1.l2_flusher

Type: [AsyncCacheFlushUnit](#).

fvp_mps2

MPS2 DUT.

Type: [FVP_MPS2](#).

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO0.busslave

Type: PVBusSlave.

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslave

Type: PVBusSlave.

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslave

Type: PVBusSlave.

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave

Type: PVBusSlave.

fvp_mps2.GPIO_connection_test

Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: GPIO1_Connection_Test.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.PSRAM.bus_slave

Type: PVBusSlave.

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.PSRAM_M7.bus_slave

Type: PVBusSlave.

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface.pvbuslave1

Type: [PVBusSlave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9**Type: [PVBusMapper](#).**fvp_mps2.clock50Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.clockdivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.cmsdk_sysctrl**

Cortex-M Simple System Control.

Type: CMSDK_SysCtrl.

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

fvp_mps2.dma1.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

fvp_mps2.dma1.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.dma1_idau_labeller

Type: `LabellerIdauSecurity`.

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: `PVBusMapper`.

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: `PVBusMaster`.

fvp_mps2.dma1_securitymodifier

Type: `SecurityModifier`.

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: `PVBusMapper`.

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: `PL080_DMACE`.

fvp_mps2.dma2.busmaster0

Type: `PVBusMaster`.

fvp_mps2.dma2.busmaster1

Type: `PVBusMaster`.

fvp_mps2.dma2.busslave

Type: `PVBusSlave`.

fvp_mps2.dma2.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: MPS2_SecureCtrl.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iod

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_iod

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: [PVBUSSlave](#).

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: [MPS2_Visualisation](#).

fvp_mps2.niden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: `PL022_SSP_MPS2`.

fvp_mps2.pl022_ssp_mps2.busslave

Type: `PVBusSlave`.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: `PVBusRouter`.

fvp_mps2.platform_bus_switch.mapper

Type: `PVBusMapper`.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: `PVBusRouter`.

fvp_mps2.platform_switch_dma0.mapper

Type: `PVBusMapper`.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: `PVBusRouter`.

fvp_mps2.platform_switch_dma1.mapper

Type: `PVBusMapper`.

fvp_mps2.signal_router

Signal router.

Type: `SignalRouter`.

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

fvp_mps2.smc_91c111.SMSC_slave

Type: `PVBusSlave`.

fvp_mps2.spiden_or_gate

Or Gate.

Type: `OrGate`.

fvp_mps2.spniden_or_gate

Or Gate.

Type: `OrGate`.

fvp_mps2.sse200

SSE-200 subsystem.

Type: SSE200.

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.crypto_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.dbg_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: `IoTSS_MessageHandlingUnit`.

fvp_mps2.sse200.mhu0.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: `IoTSS_MessageHandlingUnit`.

fvp_mps2.sse200.mhu1.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.s32k_timer.busslave

Type: PVBusSlave.

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.sse200.s32k_watchdog.busslave

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: MPS2_SecureCtrl.

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: PVBusMaster.

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.sse200.secure_watchdog.busslave

Type: PVBusSlave.

fvp_mps2.sse200.signal_router

Signal router.

Type: `SignalRouter`.

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.sys_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.sse200.timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.sse200.timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.ssram1.bus_slave

Type: `PVBusSlave`.

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).

fvp_mps2.touchscreen_interface.pvbusslaveType: [PVBusSlave](#).**fvp_mps2.uart_overflows_or_gate**

Or Gate.

Type: [OrGate](#).**idau.bus_bridge**

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.7 FVP_MPS2_Cortex-M35P

List of instances in FVP_MPS2_Cortex-M35P.

About FVP_MPS2_Cortex-M35P

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_Cortex-M35P instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**clk25khz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**cpu0**

ARM CORTEXM35P CT model.

Type: [ARM_Cortex-M35P](#).**cpu0.acp_mapper**Type: [PVBusMapper](#).**cpu0.ext_bus**Type: [PVBusLogger](#).

cpu0.ext_bus.mapperType: [PVBusMapper](#).**cpu0.l2_flusher**

Type: AsyncCacheFlushUnit.

cpu1

ARM CORTEXM35P CT model.

Type: [ARM_Cortex-M35P](#).**cpu1.acp_mapper**Type: [PVBusMapper](#).**cpu1.ext_bus**Type: [PVBusLogger](#).**cpu1.ext_bus.mapper**Type: [PVBusMapper](#).**cpu1.l2_flusher**

Type: AsyncCacheFlushUnit.

fvp_mps2

MPS2 DUT.

Type: FVP_MPS2.

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO0.busslaveType: [PVBusSlave](#).**fvp_mps2.GPIO1**

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslaveType: [PVBusSlave](#).**fvp_mps2.GPIO2**

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslaveType: [PVBusSlave](#).**fvp_mps2.GPIO3**

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslaveType: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test

Type: `GPIO_Connection_Test`.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: `GPIO_Port_Transfer`.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: `GPIO1_Connection_Test`.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM.bus_slave

Type: `PVBusSlave`.

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM_M7.bus_slave

Type: `PVBusSlave`.

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.UART0.pvbusslave

Type: `PVBusSlave`.

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: `CMSDK_UART`.

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.UART1.pvbusslave

Type: `PVBusSlave`.

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.UART2.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbuslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: `SecurityModifier`.

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: `PVBusMapper`.

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: `PL080_DMAC`.

fvp_mps2.dma2.busmaster0

Type: `PVBusMaster`.

fvp_mps2.dma2.busmaster1

Type: `PVBusMaster`.

fvp_mps2.dma2.busslave

Type: `PVBusSlave`.

fvp_mps2.dma2.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread`.

fvp_mps2.dma2.timer.timer

A `ClockTimerThread64` is a drop-in replacement for `ClockTimer64`. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

fvp_mps2.dma2.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

fvp_mps2.dma2.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.dma2_idau_labeller

Type: `LabellerIdauSecurity`.

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller
Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster
Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier
Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster
Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier
Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: PVBusSlave.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: PVBusExclusiveMonitor.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: PVBusExclusiveMonitor.

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: IoTSS_MemoryProtectionController.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: PVBusMapper.

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: [PVBusSlave](#).

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_bus_switch.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cordio_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu0core_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu0core_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu0dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu0dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu1core_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu1core_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.cpu1dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.cpu1dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.crypto_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.crypto_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.dbg_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.dbg_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0**

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: PVBusSlave.

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram1.bus_slave

Type: PVBusSlave.

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram2.bus_slave

Type: PVBusSlave.

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub0.bus_slave

Type: PVBusSlave.

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub1.bus_slave

Type: PVBusSlave.

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub_i2c1.bus_slave

Type: PVBusSlave.

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).

fvp_mps2.touchscreen_interface.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.uart_overflows_or_gate

Or Gate.

Type: [OrGate](#).

idau.bus_bridge

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.8 FVP_MPS2_Cortex-M4

List of instances in FVP_MPS2_Cortex-M4.

FVP_MPS2_Cortex-M4 instances

armcortexm4ct

ARM CORTEXM4 CT model.

Type: [ARM_Cortex-M4](#).

armcortexm4ct.acp_mapper

Type: [PVBusMapper](#).

armcortexm4ct.ext_bus

Type: [PVBusLogger](#).

armcortexm4ct.ext_bus.mapper

Type: [PVBusMapper](#).

armcortexm4ct.l2_flusher

Type: [AsyncCacheFlushUnit](#).

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2

MPS2 DUT.

Type: [FVP_MPS2](#).

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO0.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO1.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO2.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave

Type: PVBusSlave.

fvp_mps2.GPIO_connection_test

Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: GPIO1_Connection_Test.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.PSRAM.bus_slave

Type: PVBusSlave.

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.PSRAM_M7.bus_slave

Type: PVBusSlave.

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.UART0.pvbusslave

Type: PVBusSlave.

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface.pvbusslave1

Type: [PVBusSlave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: SecurityModifier.

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: PVBusMapper.

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: PL080_DMAC.

fvp_mps2.dma1.busmaster0

Type: PVBusMaster.

fvp_mps2.dma1.busmaster1

Type: PVBusMaster.

fvp_mps2.dma1.busslave

Type: PVBusSlave.

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: ClockTimerThread.

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: ClockTimerThread64.

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: SchedulerThread.

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: SchedulerThreadEvent.

fvp_mps2.dma1_idau_labeller

Type: LabellerIdauSecurity.

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: PVBusMaster.

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: `MPS2_Audio`.

fvp_mps2.mps2_audio.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: `CMSDK_DualTimer`.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: `MPS2_LCD`.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.mps2_timer1.busslave

Type: PVBusSlave.

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBusSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cordio_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.crypto_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: `IoTSS_CPUIdentity`.

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: `IoTSS_SystemControl`.

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: `PVBusMaster`.

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: `IoTSS_SystemInfo`.

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2c1
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_i2s
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_spi0
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_spi2
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer
Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0
ARM Dual-Timer Module.
Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1
Internal component used by SP804 Timer module.
Type: CounterModule.

fvp_mps2.switch_PSRAM_M7
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.switch_PSRAM_M7.mapper
Type: PVBusMapper.

fvp_mps2.switch_svos_dualtimer
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.switch_svos_dualtimer.mapper
Type: PVBusMapper.

fvp_mps2.telnetterminal0
Telnet terminal interface.
Type: TelnetTerminal.

fvp_mps2.telnetterminal1
Telnet terminal interface.
Type: TelnetTerminal.

fvp_mps2.telnetterminal2
Telnet terminal interface.
Type: TelnetTerminal.

fvp_mps2.touchscreen_interface
MPS2 Touch Screen.
Type: MPS2_TouchScreen.

fvp_mps2.touchscreen_interface.pvbusslave
Type: PVBusSlave.

fvp_mps2.uart_overflows_or_gate
Or Gate.
Type: OrGate.

15.9 FVP_MPS2_Cortex-M52

List of instances in FVP_MPS2_Cortex-M52.

About FVP_MPS2_Cortex-M52

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of

these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_Cortex-M52 has the following limitations:

- It does not have the Random number generator or Unique ID/secure storage mentioned in the MPS2 specification.
- The Ethernet controller in the model is a LAN91C111. The MPS2 documents, including those for pre-v8-M cores, specify a LAN9220.
- MTB, ETM, and TPIU are not supported. MTB RAM is absent.
- In the Memory Gating Unit, the model provides a configurable block size. For performance reasons, the minimum block size in the model is 4096 bytes. Hardware and later models may allow smaller block sizes. Software should always use the `BLK_CFG` register to determine block size.
- As in other MPS2 Fast Models, some of the peripherals have minimal implementations:
 - The Audio controller is RAZ/WI.
 - Only the touchscreen functionality of the STMPE811 touchscreen controller is implemented.
 - A subset of the Ampire LCD module's graphics modes are supported.
- See the full set of model parameters and their descriptions by running the model with the `--list-params` argument. Most parameters share their names with a corresponding RTL configuration parameter.

FVP_MPS2_Cortex-M52 instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

core0_bus_gasket

STLBusGasket allows a debugger or emulated T32 code to force the results of system-register reads by writing an address to the ADDR register then 32-bit values to the VALUE register, which are placed in a fifo associated with that address. A PVBUS transaction into pvbus_in goes unchanged to pvbus_out, unless its address matches that associated with a

non-empty fifo, in which case: writes are ignored, non-word reads abort, and word reads take values from that fifo.

Type: [STLBusGasket](#).

core0_bus_gasket.busmapper

Type: [PVBusMapper](#).

cpu0

ARM Cortex-M52 CT model.

Type: [ARM_Cortex-M52](#).

cpu0.acp_mapper

Type: [PVBusMapper](#).

cpu0.ext_bus

Type: [PVBusLogger](#).

cpu0.ext_bus.mapper

Type: [PVBusMapper](#).

cpu0.l1_incoherent_interconnect

Type: [PVCache](#).

cpu0.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[2]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu0.l1dcache
Type: [PVCACHE](#).

cpu0.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu0.l1icache
Type: [PVCACHE](#).

cpu0.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cpu0.l2_flusher
Type: [ASYNCCACHEFLUSHUNIT](#).

cpu1
ARM Cortex-M52 CT model.
Type: [ARM_Cortex-M52](#).

cpu1.acp_mapper
Type: [PVBUSMAPPER](#).

cpu1.ext_bus
Type: [PVBUSLOGGER](#).

cpu1.ext_bus.mapper
Type: [PVBUSMAPPER](#).

cpu1.l1_incoherent_interconnect
Type: [PVCACHE](#).

cpu1.l1_incoherent_interconnect.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[10]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[11]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[12]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[13]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[14]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[15]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[16]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[17]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[1]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[2]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu1.l1dcache
Type: [PVCACHE](#).

cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1icacheType: `PVCache`.**cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cpu1.l2_flusher**Type: `AsyncCacheFlushUnit`.**fvp_mps2**

MPS2 DUT.

Type: `FVP_MPS2`.**fvp_mps2.GPIO0**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO0.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO1**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO1.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO2**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO2.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO3**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO3.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO_connection_test**Type: `GPIO_Connection_Test`.**fvp_mps2.GPIO_connection_test.GPIO0_port_trans**Type: `GPIO_Port_Transfer`.**fvp_mps2.GPIO_connection_test.GPIO1_port_test**Type: `GPIO1_Connection_Test`.**fvp_mps2.PSRAM**

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbusslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbusslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion2
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMACE](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_securitymodifier
Type: SecurityModifier.

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_squasher
Squashes the exclusive attribute on bus transactions.
Type: PVBusExclusiveSquasher.

fvp_mps2.exclusive_squasher.bus_modifier
Type: PVBusMapper.

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBUSSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBUSSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBUSMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: PVBUSMapper.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_switch_dma1.mapper

Type: PVBUSMapper.

fvp_mps2.signal_router

Signal router.

Type: SignalRouter.

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.crypto_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [SVOS_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal1**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal2**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.touchscreen_interface**

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).**fvp_mps2.touchscreen_interface.pvbuslave**Type: [PVBusSlave](#).**fvp_mps2.uart_overflows_or_gate**

Or Gate.

Type: [OrGate](#).**idau.bus_bridge**

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.10 FVP_MPS2_Cortex-M55

List of instances in FVP_MPS2_Cortex-M55.

About FVP_MPS2_Cortex-M55

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_Cortex-M55 has the following differences from the MPS2 specification:

- The model does not have the random number generator or unique ID/secure storage mentioned in the MPS2 specification because the programmer's view of these devices is yet to be defined.
- The Ethernet controller in the model is a LAN91C111. The MPS2 documents, including those for pre-v8-M cores, specify a LAN9220.
- MTB, ETM, and TPIU are not supported. MTB RAM is absent.

- In the Memory Gating Unit, the model provides a configurable block size. For performance reasons, the minimum block size in the model is 4096 bytes. Hardware and later models might allow smaller block sizes. Software should always use the `BLK_CFG` register to determine block size.
- As in other MPS2 Fast Models, some of the peripherals have minimal implementations:
 - The Audio controller is RAZ/WI.
 - Only the touchscreen functionality of the STMPE811 touchscreen controller is implemented.
 - A subset of the Ampire LCD module's graphics modes are supported.

FVP_MPS2_Cortex-M55 instances

`clk25Mhz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`clk25khz`

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

`core0_bus_gasket`

STLBusGasket allows a debugger or emulated T32 code to force the results of system-register reads by writing an address to the ADDR register then 32-bit values to the VALUE register, which are placed in a fifo associated with that address. A PVBUS transaction into `pvbus_in` goes unchanged to `pvbus_out`, unless its address matches that associated with a non-empty fifo, in which case: writes are ignored, non-word reads abort, and word reads take values from that fifo.

Type: [STLBusGasket](#).

`core0_bus_gasket.busmapper`

Type: [PVBUSMapper](#).

`cpu0`

ARM Cortex-M55 CT model.

Type: [ARM_Cortex-M55](#).

`cpu0.acp_mapper`

Type: [PVBUSMapper](#).

`cpu0.ext_bus`

Type: [PVBUSLogger](#).

`cpu0.ext_bus.mapper`

Type: [PVBUSMapper](#).

cpu0.l1_incoherent_interconnect
Type: [PVCache](#).

cpu0.l1_incoherent_interconnect.upstream[0]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[10]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[11]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[12]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[13]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[14]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[15]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[16]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[17]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[1]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[2]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[3]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[4]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[5]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[6]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[7]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[8]
Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[9]
Type: [PVBusSlave](#).

cpu0.l1dcache

Type: [PVCache](#).

cpu0.l1dcache.upstream[0]

Type: [PVBusSlave](#).

cpu0.l1icache

Type: [PVCache](#).

cpu0.l1icache.upstream[0]

Type: [PVBusSlave](#).

cpu0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cpu1

ARM Cortex-M55 CT model.

Type: [ARM_Cortex-M55](#).

cpu1.acp_mapper

Type: [PVBusMapper](#).

cpu1.ext_bus

Type: [PVBusLogger](#).

cpu1.ext_bus.mapper

Type: [PVBusMapper](#).

cpu1.l1_incoherent_interconnect

Type: [PVCache](#).

cpu1.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

cpu1.l1_incoherent_interconnect.upstream[17]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[1]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[2]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu1.l1dcache
Type: [PVCACHE](#).

cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1icache
Type: [PVCACHE](#).

cpu1.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l2_flusher
Type: [ASYNCCACHEFLUSHUNIT](#).

fvp_mps2
MPS2 DUT.
Type: [FVP_MPS2](#).

fvp_mps2.GPIO0
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO0.busslave
Type: [PVBUSSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test

Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: GPIO1_Connection_Test.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusslave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusslave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusslave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusslave](#).

fvp_mps2.VGA_interface.pvbusslave1

Type: [PVBusslave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8**Type: [PVBusMapper](#).**fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9**Type: [PVBusMapper](#).**fvp_mps2.clock50Hz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.clockdivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.cmsdk_sysctrl**

Cortex-M Simple System Control.

Type: CMSDK_SysCtrl.

fvp_mps2.cmsdk_sysctrl.busslaveType: [PVBusSlave](#).**fvp_mps2.cmsdk_watchdog**

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.cmsdk_watchdog.busslaveType: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mpc_iotss_ssram3.busslave

Type: `PVBusSlave`.

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.mps2_audio

MPS2 Audio.

Type: `MPS2_Audio`.

fvp_mps2.mps2_audio.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: `CMSDK_DualTimer`.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: [MPS2_LCD](#).

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.mps2_timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: [PVBusSlave](#).

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_bus_switch.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMSC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [sse200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_cpu0.bus_mapper
Type: `PVBusMapper`.

fvp_mps2.sse200.acg_cpu1
IoT Subsystem Access Control Gate.
Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_cpu1.bus_mapper
Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram0
IoT Subsystem Access Control Gate.
Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram0.bus_mapper
Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram1
IoT Subsystem Access Control Gate.
Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram1.bus_mapper
Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram2
IoT Subsystem Access Control Gate.
Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram2.bus_mapper
Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram3
IoT Subsystem Access Control Gate.
Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram3.bus_mapper
Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0
Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1
Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10
Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11
Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8**Type: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9**Type: [PVBusMapper](#).**fvp_mps2.sse200.clock32kHz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.sse200.clockdivider**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.sse200.cmsdk_dualtimer**

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).**fvp_mps2.sse200.cmsdk_dualtimer.busslave**Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.crypto_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: MPS2_SecureCtrl.

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: SignalRouter.

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer0.busslave

Type: PVBusSlave.

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: PVBusSlave.

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram1.bus_slave

Type: PVBusSlave.

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.ssram2.bus_slave

Type: PVBusSlave.

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: RAMDevice.

fvp_mps2.stub0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [SVOS_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: MPS2_TouchScreen.

fvp_mps2.touchscreen_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.uart_overflows_or_gate

Or Gate.

Type: [OrGate](#).

idau.bus_bridge

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.11 FVP_MPS2_Cortex-M7

List of instances in FVP_MPS2_Cortex-M7.

FVP_MPS2_Cortex-M7 instances

armcortexm7ct

ARM CORTEXM7 CT model.

Type: [ARM_Cortex-M7](#).

armcortexm7ct.acp_mapper

Type: [PVBusMapper](#).

armcortexm7ct.ext_bus

Type: [PVBusLogger](#).

armcortexm7ct.ext_bus.mapper

Type: [PVBusMapper](#).

armcortexm7ct.l1_incoherent_interconnect

Type: [PVCache](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[2]

Type: [PVBusSlave](#).

armcortexm7ct.l1_incoherent_interconnect.upstream[3]Type: [PVBUSSlave](#).**armcortexm7ct.l1_incoherent_interconnect.upstream[4]**Type: [PVBUSSlave](#).**armcortexm7ct.l1_incoherent_interconnect.upstream[5]**Type: [PVBUSSlave](#).**armcortexm7ct.l1_incoherent_interconnect.upstream[6]**Type: [PVBUSSlave](#).**armcortexm7ct.l1_incoherent_interconnect.upstream[7]**Type: [PVBUSSlave](#).**armcortexm7ct.l1_incoherent_interconnect.upstream[8]**Type: [PVBUSSlave](#).**armcortexm7ct.l1_incoherent_interconnect.upstream[9]**Type: [PVBUSSlave](#).**armcortexm7ct.l1dcache**Type: [PVCACHE](#).**armcortexm7ct.l1dcache.upstream[0]**Type: [PVBUSSlave](#).**armcortexm7ct.l1icache**Type: [PVCACHE](#).**armcortexm7ct.l1icache.upstream[0]**Type: [PVBUSSlave](#).**armcortexm7ct.l2_flusher**Type: [ASYNCCACHEFLUSHUNIT](#).**clk25Mhz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**clk25khz**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2**

MPS2 DUT.

Type: [FVP_MPS2](#).

fvp_mps2.GPIO0

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO0.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO1

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO2

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO3

ARM PrimeCell General Purpose Input/Output.

Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test

Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: GPIO1_Connection_Test.

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: [MPS2_VGA](#).

fvp_mps2.VGA_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface.pvbusslave1

Type: [PVBusSlave](#).

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: CMSDK_SysCtrl.

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a

proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: `ClockTimerThread64`.

fvp_mps2.dma1.timer.timer.thread

A `SchedulerThread` instance represents a co-routine thread in the simulation.

Type: `SchedulerThread`.

fvp_mps2.dma1.timer.timer.thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.dma1_idau_labeller

Type: `LabellerIdauSecurity`.

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: `PVBusMapper`.

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: `PVBusMaster`.

fvp_mps2.dma1_securitymodifier

Type: `SecurityModifier`.

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: `PVBusMapper`.

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: `PL080_DMACE`.

fvp_mps2.dma2.busmaster0

Type: `PVBusMaster`.

fvp_mps2.dma2.busmaster1

Type: `PVBusMaster`.

fvp_mps2.dma2.busslave

Type: `PVBusSlave`.

fvp_mps2.dma2.timer

A `ClockTimerThread(64)` is a drop-in replacement for a `ClockTimer(64)` component. The main difference to the `ClockTimer` component is that the `ClockTimerThread` runs the `signal()` callback from a proper scheduler thread. This mean that the `signal()` function may directly or indirectly invoke `wait()` functions to wait for time or events. This is not allowed for the `ClockTimer` component which does not use a thread. Components which issue bus transactions from within the timer `signal()` callback must use `ClockTimerThread(64)` rather than `ClockTimer(64)`.

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal()

callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: MPS2_SecureCtrl.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iod

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_iod

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.mps2_timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: [MPS2_Visualisation](#).

fvp_mps2.niden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: `PL022_SSP_MPS2`.

fvp_mps2.pl022_ssp_mps2.busslave

Type: `PVBusSlave`.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: `PVBusRouter`.

fvp_mps2.platform_bus_switch.mapper

Type: `PVBusMapper`.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: `PVBusRouter`.

fvp_mps2.platform_switch_dma0.mapper

Type: `PVBusMapper`.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: `PVBusRouter`.

fvp_mps2.platform_switch_dma1.mapper

Type: `PVBusMapper`.

fvp_mps2.signal_router

Signal router.

Type: `SignalRouter`.

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: `SMSC_91C111`.

fvp_mps2.smc_91c111.SMSC_slave

Type: `PVBusSlave`.

fvp_mps2.spiden_or_gate

Or Gate.

Type: `OrGate`.

fvp_mps2.spniden_or_gate

Or Gate.

Type: `OrGate`.

fvp_mps2.sse200

SSE-200 subsystem.

Type: SSE200.

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_cpu0.bus_mapperType: [PVBusMapper](#).**fvp_mps2.sse200.acg_cpu1**

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_cpu1.bus_mapperType: [PVBusMapper](#).**fvp_mps2.sse200.acg_sram0**

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram0.bus_mapperType: [PVBusMapper](#).**fvp_mps2.sse200.acg_sram1**

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram1.bus_mapperType: [PVBusMapper](#).**fvp_mps2.sse200.acg_sram2**

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram2.bus_mapperType: [PVBusMapper](#).**fvp_mps2.sse200.acg_sram3**

IoT Subsystem Access Control Gate.

Type: IoTSS_AccessControlGate.

fvp_mps2.sse200.acg_sram3.bus_mapperType: [PVBusMapper](#).**fvp_mps2.sse200.apb_ppc_iotss_subsystem0**

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A [ClockDivider](#) is a library component that takes a [ClockSignal](#) on its input port (which could come from the output of a [MasterClock](#), or from another [ClockDivider](#)), and generates a new [ClockSignal](#) on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.crypto_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.dbg_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: `IoTSS_MessageHandlingUnit`.

fvp_mps2.sse200.mhu0.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: `IoTSS_MessageHandlingUnit`.

fvp_mps2.sse200.mhu1.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.s32k_timer.busslave

Type: PVBusSlave.

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.sse200.s32k_watchdog.busslave

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: MPS2_SecureCtrl.

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: PVBusMapper.

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: PVBusSlave.

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: PVBusMaster.

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: CMSDK_Watchdog.

fvp_mps2.sse200.secure_watchdog.busslave

Type: PVBusSlave.

fvp_mps2.sse200.signal_router

Signal router.

Type: `SignalRouter`.

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: `PPUv0`.

fvp_mps2.sse200.sys_ppu.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.sse200.timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.sse200.timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.ssram1.bus_slave

Type: `PVBusSlave`.

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).

fvp_mps2.touchscreen_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.uart_overflows_or_gate

Or Gate.

Type: [OrGate](#).

15.12 FVP_MPS2_Cortex-M85

List of instances in FVP_MPS2_Cortex-M85.

About FVP_MPS2_Cortex-M85

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_Cortex-M85 has the following differences from the MPS2 specification:

- The model does not have the random number generator or unique ID/secure storage mentioned in the MPS2 specification because the programmer's view of these devices is yet to be defined.
- The Ethernet controller in the model is a LAN91C111. The MPS2 documents, including those for pre-v8-M cores, specify a LAN9220.
- MTB, ETM, and TPIU are not supported. MTB RAM is absent.
- In the Memory Gating Unit, the model provides a configurable block size. For performance reasons, the minimum block size in the model is 4096 bytes. Hardware and later models might allow smaller block sizes. Software should always use the `BLK_CFG` register to determine block size.
- As in other MPS2 Fast Models, some of the peripherals have minimal implementations:
 - The Audio controller is RAZ/WI.
 - Only the touchscreen functionality of the STMPE811 touchscreen controller is implemented.
 - A subset of the Ampire LCD module's graphics modes are supported.
- See the full set of model parameters and their descriptions by running the model with the `--list-params` argument. Most parameters share their names with a corresponding RTL configuration parameter.

FVP_MPS2_Cortex-M85 instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

core0_bus_gasket

STLBusGasket allows a debugger or emulated T32 code to force the results of system-register reads by writing an address to the ADDR register then 32-bit values to the VALUE register, which are placed in a fifo associated with that address. A PVBUS transaction into pvbus_in goes unchanged to pvbus_out, unless its address matches that associated with a non-empty fifo, in which case: writes are ignored, non-word reads abort, and word reads take values from that fifo.

Type: [STLBusGasket](#).

core0_bus_gasket.busmapper

Type: [PVBUSMapper](#).

cpu0

ARM Cortex-M85 CT model.

Type: [ARM_Cortex-M85](#).

cpu0.acp_mapper

Type: [PVBUSMapper](#).

cpu0.ext_bus

Type: [PVBUSLogger](#).

cpu0.ext_bus.mapper

Type: [PVBUSMapper](#).

cpu0.l1_incoherent_interconnect

Type: [PVCache](#).

cpu0.l1_incoherent_interconnect.upstream[0]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[10]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[11]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[12]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[13]

Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[14]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[15]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[16]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[17]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[1]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[2]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu0.l1dcache
Type: [PVCACHE](#).

cpu0.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu0.l1icache
Type: [PVCACHE](#).

cpu0.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cpu0.l2_flusher
Type: [ASYNCCACHEFLUSHUNIT](#).

cpu1
ARM Cortex-M85 CT model.

Type: `ARM_Cortex-M85`.

`cpu1.acp_mapper`
Type: `PVBusMapper`.

`cpu1.ext_bus`
Type: `PVBusLogger`.

`cpu1.ext_bus.mapper`
Type: `PVBusMapper`.

`cpu1.l1_incoherent_interconnect`
Type: `PVCache`.

`cpu1.l1_incoherent_interconnect.upstream[0]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[10]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[11]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[12]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[13]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[14]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[15]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[16]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[17]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[1]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[2]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[3]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[4]`
Type: `PVBusSlave`.

`cpu1.l1_incoherent_interconnect.upstream[5]`
Type: `PVBusSlave`.

cpu1.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu1.l1dcache
Type: [PVCACHE](#).

cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1icache
Type: [PVCACHE](#).

cpu1.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l2_flusher
Type: [ASYNCCACHEFLUSHUNIT](#).

fvp_mps2
MPS2 DUT.
Type: [FVP_MPS2](#).

fvp_mps2.GPIO0
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO0.busslave
Type: [PVBUSSlave](#).

fvp_mps2.GPIO1
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO1.busslave
Type: [PVBUSSlave](#).

fvp_mps2.GPIO2
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO2.busslave
Type: [PVBUSSlave](#).

fvp_mps2.GPIO3
ARM PrimeCell General Purpose Input/Output.
Type: [CMSDK_GPIO](#).

fvp_mps2.GPIO3.busslave

Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test

Type: [GPIO_Connection_Test](#).

fvp_mps2.GPIO_connection_test.GPIO0_port_trans

Type: [GPIO_Port_Transfer](#).

fvp_mps2.GPIO_connection_test.GPIO1_port_test

Type: [GPIO1_Connection_Test](#).

fvp_mps2.PSRAM

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: CMSDK_UART.

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.UART2.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbuslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmasterType: [PVBusMaster](#).**fvp_mps2.dma1_securitymodifier**Type: [SecurityModifier](#).**fvp_mps2.dma1_securitymodifier.pvbusmodifier**Type: [PVBusMapper](#).**fvp_mps2.dma2**

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).**fvp_mps2.dma2.busmaster0**Type: [PVBusMaster](#).**fvp_mps2.dma2.busmaster1**Type: [PVBusMaster](#).**fvp_mps2.dma2.busslave**Type: [PVBusSlave](#).**fvp_mps2.dma2.timer**

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).**fvp_mps2.dma2.timer.timer**

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).**fvp_mps2.dma2.timer.timer.thread**A [SchedulerThread](#) instance represents a co-routine thread in the simulation.Type: [SchedulerThread](#).**fvp_mps2.dma2.timer.timer.thread_event**

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).**fvp_mps2.dma2_idau_labeller**Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma3_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_psram_iotss

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBusMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: `MPS2_Audio`.

fvp_mps2.mps2_audio.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: `CMSDK_DualTimer`.

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: `PVBusExclusiveMonitor`.

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: `MPS2_LCD`.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: `IoTSS_MemoryProtectionController`.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A `SchedulerThreadEvent` instance is an auto-reset boolean condition variable other threads can wait on.

Type: `SchedulerThreadEvent`.

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.mps2_timer1.busslave

Type: PVBusSlave.

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBusSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: [PVBusMapper](#).

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.platform_switch_dma1.mapper

Type: [PVBusMapper](#).

fvp_mps2.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.smsc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smsc_91c111.SMSC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: `IoTSS_AccessControlGate`.

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: `PVBusMapper`.

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cordio_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.crypto_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: PPUv0.

fvp_mps2.sse200.dbg_ppu.busslave

Type: PVBusSlave.

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: `IoTSS_CPUIdentity`.

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: `IoTSS_SystemControl`.

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: `PVBusMaster`.

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: `IoTSS_SystemInfo`.

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: `PVBusSlave`.

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iot

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: CMSDK_Timer.

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_i2c1
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_i2s
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_spi0
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.stub_spi2
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer
Type: [svos_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3
Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0
ARM Dual-Timer Module.
Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave
Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1
Internal component used by SP804 Timer module.
Type: CounterModule.

fvp_mps2.switch_PSRAM_M7
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.switch_PSRAM_M7.mapper
Type: PVBusMapper.

fvp_mps2.switch_svos_dualtimer
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.switch_svos_dualtimer.mapper
Type: PVBusMapper.

fvp_mps2.telnetterminal0
Telnet terminal interface.
Type: TelnetTerminal.

fvp_mps2.telnetterminal1
Telnet terminal interface.
Type: TelnetTerminal.

fvp_mps2.telnetterminal2
Telnet terminal interface.
Type: TelnetTerminal.

fvp_mps2.touchscreen_interface
MPS2 Touch Screen.
Type: MPS2_TouchScreen.

fvp_mps2.touchscreen_interface.pvbusslave
Type: PVBusSlave.

fvp_mps2.uart_overflows_or_gate
Or Gate.
Type: OrGate.

idau.bus_bridge
Bridge incoming transactions to a PVDevice port.
Type: PVBusBridge.

15.13 FVP_MPS2_SSE-200_Cortex-M33

List of instances in FVP_MPS2_SSE-200_Cortex-M33.

About FVP_MPS2_SSE-200_Cortex-M33

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \  
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_SSE-200_Cortex-M33 instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cpu0

ARM CORTEXM33 CT model.

Type: [ARM_Cortex-M33](#).

cpu0.acp_mapper

Type: [PVBusMapper](#).

cpu0.ext_bus

Type: [PVBusLogger](#).

cpu0.ext_bus.mapper

Type: [PVBusMapper](#).

cpu0.l2_flusher

Type: [AsyncCacheFlushUnit](#).

cpu1

ARM CORTEXM33 CT model.

Type: [ARM_Cortex-M33](#).

cpu1.acp_mapper

Type: [PVBusMapper](#).

cpu1.ext_bus
Type: [PVBusLogger](#).

cpu1.ext_bus.mapper
Type: [PVBusMapper](#).

cpu1.l2_flusher
Type: AsyncCacheFlushUnit.

fvp_mps2
MPS2 DUT.
Type: FVP_MPS2.

fvp_mps2.GPIO0
ARM PrimeCell General Purpose Input/Output.
Type: CMSDK_GPIO.

fvp_mps2.GPIO0.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO1
ARM PrimeCell General Purpose Input/Output.
Type: CMSDK_GPIO.

fvp_mps2.GPIO1.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO2
ARM PrimeCell General Purpose Input/Output.
Type: CMSDK_GPIO.

fvp_mps2.GPIO2.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO3
ARM PrimeCell General Purpose Input/Output.
Type: CMSDK_GPIO.

fvp_mps2.GPIO3.busslave
Type: [PVBusSlave](#).

fvp_mps2.GPIO_connection_test
Type: GPIO_Connection_Test.

fvp_mps2.GPIO_connection_test.GPIO0_port_trans
Type: GPIO_Port_Transfer.

fvp_mps2.GPIO_connection_test.GPIO1_port_test
Type: GPIO1_Connection_Test.

fvp_mps2.PSRAM
RAM device, can be dynamic or static ram.
Type: [RAMDevice](#).

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbuslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbuslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion2
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMACE](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBUSMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBUSSlave](#).

fvp_mps2.dma3.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBUSMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBUSMaster](#).

fvp_mps2.dma3_securitymodifier
Type: SecurityModifier.

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_squasher
Squashes the exclusive attribute on bus transactions.
Type: PVBusExclusiveSquasher.

fvp_mps2.exclusive_squasher.bus_modifier
Type: PVBusMapper.

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbusslave

Type: [PVBUSSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer1.clk_div

A `ClockDivider` is a library component that takes a `ClockSignal` on its input port (which could come from the output of a `MasterClock`, or from another `ClockDivider`), and generates a new `ClockSignal` on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBusSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: PVBusMapper.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: PVBusRouter.

fvp_mps2.platform_switch_dma1.mapper

Type: PVBusMapper.

fvp_mps2.signal_router

Signal router.

Type: SignalRouter.

fvp_mps2.smc_91c111
SMSC 91C111 ethernet controller.
Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC slave
Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate
Or Gate.
Type: [OrGate](#).

fvp_mps2.spniden_or_gate
Or Gate.
Type: [OrGate](#).

fvp_mps2.sse200
SSE-200 subsystem.
Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2
IoT Subsystem Access Control Gate.
Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper
Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.crypto_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.ram2_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.ram3_ppu**

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).**fvp_mps2.sse200.ram3_ppu.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.s32k_timer**

ARM Timer Module.

Type: [CMSDK_Timer](#).**fvp_mps2.sse200.s32k_timer.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.s32k_timer.clk_div**

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).**fvp_mps2.sse200.s32k_timer.counter**

Internal component used by SP804 Timer module.

Type: [CounterModule](#).**fvp_mps2.sse200.s32k_watchdog**

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).**fvp_mps2.sse200.s32k_watchdog.busslave**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block**

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).**fvp_mps2.sse200.secure_control_register_block.bus_mapper**Type: [PVBusMapper](#).**fvp_mps2.sse200.secure_control_register_block.busslave_ns_iot**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2**Type: [PVBusSlave](#).**fvp_mps2.sse200.secure_control_register_block.busslave_s_iot**Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [SVOS_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal1**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.telnetterminal2**

Telnet terminal interface.

Type: [TelnetTerminal](#).**fvp_mps2.touchscreen_interface**

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).**fvp_mps2.touchscreen_interface.pvbuslave**Type: [PVBusSlave](#).**fvp_mps2.uart_overflows_or_gate**

Or Gate.

Type: [OrGate](#).**idau.bus_bridge**

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

15.14 FVP_MPS2_SSE-200_Cortex-M55

List of instances in FVP_MPS2_SSE-200_Cortex-M55.

About FVP_MPS2_SSE-200_Cortex-M55

The MPS2 platform includes many features that enable the creation of complex software. For those new to TrustZone® for Cortex®-M, however, it might be useful to run the model with some of these features disabled. Running the model with the following parameters disables the IDAU and the Security Gates:

```
--parameter idau.NUM_IDAU_REGION=0 \
--parameter fvp_mps2.DISABLE_GATING=1
```

FVP_MPS2_SSE-200_Cortex-M55 instances

clk25Mhz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

clk25khz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

cpu0

ARM Cortex-M55 CT model.

Type: [ARM_Cortex-M55](#).

cpu0.acp_mapper

Type: [PVBusMapper](#).

cpu0.ext_bus

Type: [PVBusLogger](#).

cpu0.ext_bus.mapper

Type: [PVBusMapper](#).

cpu0.l1_incoherent_interconnect

Type: [PVCache](#).

cpu0.l1_incoherent_interconnect.upstream[0]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[10]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[11]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[12]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[13]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[14]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[15]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[16]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[17]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[1]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[2]

Type: [PVBusSlave](#).

cpu0.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu0.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu0.l1dcache
Type: [PVCACHE](#).

cpu0.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu0.l1icache
Type: [PVCACHE](#).

cpu0.l1icache.upstream[0]
Type: [PVBUSSlave](#).

cpu0.l2_flusher
Type: [ASYNCCACHEFLUSHUNIT](#).

cpu1
ARM Cortex-M55 CT model.
Type: [ARM_Cortex-M55](#).

cpu1.acp_mapper
Type: [PVBUSMAPPER](#).

cpu1.ext_bus
Type: [PVBUSLOGGER](#).

cpu1.ext_bus.mapper
Type: [PVBUSMAPPER](#).

cpu1.l1_incoherent_interconnect
Type: [PVCACHE](#).

cpu1.l1_incoherent_interconnect.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[10]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[11]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[12]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[13]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[14]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[15]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[16]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[17]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[1]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[2]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[3]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[4]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[5]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[6]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[7]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[8]
Type: [PVBUSSlave](#).

cpu1.l1_incoherent_interconnect.upstream[9]
Type: [PVBUSSlave](#).

cpu1.l1dcache
Type: [PVCACHE](#).

cpu1.l1dcache.upstream[0]
Type: [PVBUSSlave](#).

cpu1.l1icacheType: `PVCache`.**cpu1.l1icache.upstream[0]**Type: `PVBusSlave`.**cpu1.l2_flusher**Type: `AsyncCacheFlushUnit`.**fvp_mps2**

MPS2 DUT.

Type: `FVP_MPS2`.**fvp_mps2.GPIO0**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO0.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO1**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO1.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO2**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO2.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO3**

ARM PrimeCell General Purpose Input/Output.

Type: `CMSDK_GPIO`.**fvp_mps2.GPIO3.busslave**Type: `PVBusSlave`.**fvp_mps2.GPIO_connection_test**Type: `GPIO_Connection_Test`.**fvp_mps2.GPIO_connection_test.GPIO0_port_trans**Type: `GPIO_Port_Transfer`.**fvp_mps2.GPIO_connection_test.GPIO1_port_test**Type: `GPIO1_Connection_Test`.**fvp_mps2.PSRAM**

RAM device, can be dynamic or static ram.

Type: `RAMDevice`.

fvp_mps2.PSRAM.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.PSRAM_M7

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.PSRAM_M7.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.UART0

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART0.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART0.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART1

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART1.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART1.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.UART2

ARM CMSDK UART Module.

Type: [CMSDK_UART](#).

fvp_mps2.UART2.clk_divider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.UART2.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.VGA_interface

VGA display interface between main bus and visualisation.

Type: MPS2_VGA.

fvp_mps2.VGA_interface.pvbuslave

Type: PVBusSlave.

fvp_mps2.VGA_interface.pvbuslave1

Type: PVBusSlave.

fvp_mps2.ahb_ppc_iotss_expansion0

IoT Subsystem Peripheral Protection Controller.

Type: IoTSS_PeripheralProtectionController.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper0

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper1

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper10

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper11

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper12

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper13

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper14

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper15

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper2

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper3

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper4

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper5

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper6

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper7

Type: PVBusMapper.

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.ahb_ppc_iotss_expansion1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion1
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper0
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper1
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper10
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper11
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper12
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper13
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper14
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper15
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper2
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper3
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper4
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper5
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper6
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper7
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper8
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion1.bus_mapper9
Type: `PVBusMapper`.

fvp_mps2.apb_ppc_iotss_expansion2
IoT Subsystem Peripheral Protection Controller.
Type: `IoTSS_PeripheralProtectionController`.

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.apb_ppc_iotss_expansion2.bus_mapper9

Type: [PVBusMapper](#).

fvp_mps2.clock50Hz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.cmsdk_sysctrl

Cortex-M Simple System Control.

Type: [CMSDK_SysCtrl](#).

fvp_mps2.cmsdk_sysctrl.busslave

Type: [PVBusSlave](#).

fvp_mps2.cmsdk_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.cmsdk_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.cpu_wait_or_gate_0

Or Gate.

Type: [OrGate](#).

fvp_mps2.cpu_wait_or_gate_1

Or Gate.

Type: [OrGate](#).

fvp_mps2.dbgen_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.dma0

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma0.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma0.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma0.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma0.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus

transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma0.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This means that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma0.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma0.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma0_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma0_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma0_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma0_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma0_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma1.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma1.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma1.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma1.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma1.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma1.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma1.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma1_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma1_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma1_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma1_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma1_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma2.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma2.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma2.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma2.timer

A [ClockTimerThread\(64\)](#) is a drop-in replacement for a [ClockTimer\(64\)](#) component. The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread](#).

fvp_mps2.dma2.timer.timer

A [ClockTimerThread64](#) is a drop-in replacement for [ClockTimer64](#). The main difference to the [ClockTimer](#) component is that the [ClockTimerThread](#) runs the [signal\(\)](#) callback from a proper scheduler thread. This mean that the [signal\(\)](#) function may directly or indirectly invoke [wait\(\)](#) functions to wait for time or events. This is not allowed for the [ClockTimer](#) component which does not use a thread. Components which issue bus transactions from within the timer [signal\(\)](#) callback must use [ClockTimerThread\(64\)](#) rather than [ClockTimer\(64\)](#).

Type: [ClockTimerThread64](#).

fvp_mps2.dma2.timer.timer.thread

A [SchedulerThread](#) instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma2.timer.timer.thread_event

A [SchedulerThreadEvent](#) instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma2_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma2_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma2_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma2_securitymodifier

Type: [SecurityModifier](#).

fvp_mps2.dma2_securitymodifier.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3

ARM PrimeCell DMA Controller(PL080/081).

Type: [PL080_DMAC](#).

fvp_mps2.dma3.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.dma3.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.dma3.busslave

Type: [PVBusSlave](#).

fvp_mps2.dma3.timer

A ClockTimerThread(64) is a drop-in replacement for a ClockTimer(64) component. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread](#).

fvp_mps2.dma3.timer.timer

A ClockTimerThread64 is a drop-in replacement for ClockTimer64. The main difference to the ClockTimer component is that the ClockTimerThread runs the signal() callback from a proper scheduler thread. This mean that the signal() function may directly or indirectly invoke wait() functions to wait for time or events. This is not allowed for the ClockTimer component which does not use a thread. Components which issue bus transactions from within the timer signal() callback must use ClockTimerThread(64) rather than ClockTimer(64).

Type: [ClockTimerThread64](#).

fvp_mps2.dma3.timer.timer.thread

A SchedulerThread instance represents a co-routine thread in the simulation.

Type: [SchedulerThread](#).

fvp_mps2.dma3.timer.timer.thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.dma3_idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.dma3_idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.dma3_idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.dma3_securitymodifier
Type: SecurityModifier.

fvp_mps2.dma3_securitymodifier.pvbusmodifier
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_psram_iotss
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_psram_iotss.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_iotss.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2
Allow transactions to be routed arbitrarily.
Type: PVBusRouter.

fvp_mps2.exclusive_monitor_switch_extra_psram_mps2.mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram1
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram1.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_monitor_zbtsram2
Global exclusive monitor.
Type: PVBusExclusiveMonitor.

fvp_mps2.exclusive_monitor_zbtsram2.bus_mapper
Type: PVBusMapper.

fvp_mps2.exclusive_squasher
Squashes the exclusive attribute on bus transactions.
Type: PVBusExclusiveSquasher.

fvp_mps2.exclusive_squasher.bus_modifier
Type: PVBusMapper.

fvp_mps2.fpga_sysctrl

FPGA SysCtrl timers LEDs and switches.

Type: [FPGA_SysCtrl](#).

fvp_mps2.fpga_sysctrl.callBack100HzCounter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.fpga_sysctrl.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.fpga_sysctrl.pvbuslave

Type: [PVBUSlave](#).

fvp_mps2.gpio_0_or_2

Or Gate.

Type: [OrGate](#).

fvp_mps2.gpio_1_or_3

Or Gate.

Type: [OrGate](#).

fvp_mps2.hostbridge

Host Socket Interface Component.

Type: [HostBridge](#).

fvp_mps2.mem_switch_extra_psram_iotss

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_iotss.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mem_switch_extra_psram_mps2

Allow transactions to be routed arbitrarily.

Type: [PVBUSRouter](#).

fvp_mps2.mem_switch_extra_psram_mps2.mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram1.bus_mapper

Type: [PVBUSMapper](#).

fvp_mps2.mpc_iotss_ssram1.busslave

Type: [PVBUSSlave](#).

fvp_mps2.mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mpc_iotss_ssram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.mpc_iotss_ssram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mpc_iotss_ssram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.mpc_iotss_ssram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_audio

MPS2 Audio.

Type: [MPS2_Audio](#).

fvp_mps2.mps2_audio.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.mps2_cmsdk_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.mps2_cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_exclusive_monitor_zbtsram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.mps2_exclusive_monitor_zbtsram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_lcd

MPS2 LCD I2C interface.

Type: MPS2_LCD.

fvp_mps2.mps2_mpc_iotss_ssram1

IoT Subsystem Memory Protection Controller.

Type: IoTSS_MemoryProtectionController.

fvp_mps2.mps2_mpc_iotss_ssram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.mps2_mpc_iotss_ssram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.mps2_mpc_iotss_ssram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.mps2_secure_control_register_block

MPS2 Secure Control Register Block.

Type: `MPS2_SecureCtrl`.

fvp_mps2.mps2_secure_control_register_block.bus_mapper

Type: `PVBusMapper`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_ns_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_iot

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.busslave_s_mps2

Type: `PVBusSlave`.

fvp_mps2.mps2_secure_control_register_block.idau_busmaster

Type: `PVBusMaster`.

fvp_mps2.mps2_timer0

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer0.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer0.counter

Internal component used by SP804 Timer module.

Type: `CounterModule`.

fvp_mps2.mps2_timer1

ARM Timer Module.

Type: `CMSDK_Timer`.

fvp_mps2.mps2_timer1.busslave

Type: `PVBusSlave`.

fvp_mps2.mps2_timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: `ClockDivider`.

fvp_mps2.mps2_timer1.counter

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.mps2_visualisation

Display window for MPS2 using Visualisation library.

Type: MPS2_Visualisation.

fvp_mps2.niden_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.nmi_or_gate

Or Gate.

Type: OrGate.

fvp_mps2.pl022_ssp_mps2

ARM PrimeCell Synchronous Serial Port(PL022).

Type: PL022_SSP_MPS2.

fvp_mps2.pl022_ssp_mps2.busslave

Type: PVBUSSlave.

fvp_mps2.pl022_ssp_mps2.prescaler

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.platform_bus_switch

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_bus_switch.mapper

Type: PVBUSMapper.

fvp_mps2.platform_switch_dma0

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_switch_dma0.mapper

Type: PVBUSMapper.

fvp_mps2.platform_switch_dma1

Allow transactions to be routed arbitrarily.

Type: PVBUSRouter.

fvp_mps2.platform_switch_dma1.mapper

Type: PVBUSMapper.

fvp_mps2.signal_router

Signal router.

Type: SignalRouter.

fvp_mps2.smc_91c111

SMSC 91C111 ethernet controller.

Type: [SMSC_91C111](#).

fvp_mps2.smc_91c111.SMC_slave

Type: [PVBusSlave](#).

fvp_mps2.spiden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.spniden_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200

SSE-200 subsystem.

Type: [SSE200](#).

fvp_mps2.sse200.acg_cpu0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_cpu1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_cpu1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram0

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram1

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram2

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.acg_sram3

IoT Subsystem Access Control Gate.

Type: [IoTSS_AccessControlGate](#).

fvp_mps2.sse200.acg_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0

IoT Subsystem Peripheral Protection Controller.

Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper0

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper1

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper10

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper11

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper12

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper13

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper14

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper15

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper2

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper3

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper4

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper5

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper6

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper7

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper8

Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem0.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1
IoT Subsystem Peripheral Protection Controller.
Type: [IoTSS_PeripheralProtectionController](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper0
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper1
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper10
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper11
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper12
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper13
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper14
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper15
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper2
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper3
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper4
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper5
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper6
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper7
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper8
Type: [PVBusMapper](#).

fvp_mps2.sse200.apb_ppc_iotss_subsystem1.bus_mapper9
Type: [PVBusMapper](#).

fvp_mps2.sse200.clock32kHz

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.clockdivider

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.sse200.cmsdk_dualtimer.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cmsdk_dualtimer.counter1

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.cordio_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cordio_ppu.busslave

Type: [PVBUSSlave](#).

fvp_mps2.sse200.cpu0core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu0dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu0dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1core_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1core_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.cpu1dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.cpu1dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.crypto_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.crypto_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.dbg_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.dbg_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3

Global exclusive monitor.

Type: [PVBusExclusiveMonitor](#).

fvp_mps2.sse200.exclusive_monitor_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.exclusive_squasher

Squashes the exclusive attribute on bus transactions.

Type: [PVBusExclusiveSquasher](#).

fvp_mps2.sse200.exclusive_squasher.bus_modifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller

Type: [LabellerIdauSecurity](#).

fvp_mps2.sse200.idau_labeller.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.idau_labeller.pvbusmodifier

Type: [PVBusMapper](#).

fvp_mps2.sse200.idau_labeller.remap_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.internal_msniper_tcm

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.internal_msniper_tcm.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_cpuidentity

IoT Subsystem CPU_IDENTITY registers.

Type: [IoTSS_CPUIdentity](#).

fvp_mps2.sse200.iotss_cpuidentity.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram1.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_internal_sram3

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.sse200.iotss_internal_sram3.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systemcontrol

IoT Subsystem System Control registers.

Type: [IoTSS_SystemControl](#).

fvp_mps2.sse200.iotss_systemcontrol.busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.iotss_systemcontrol.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.iotss_systeminfo

IoT Subsystem System Information registers.

Type: [IoTSS_SystemInfo](#).

fvp_mps2.sse200.iotss_systeminfo.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_msniper_tcm.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_internal_sram_mpc.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mem_switch_ppu

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.sse200.mem_switch_ppu.mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mhu0

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mhu1

IoT Subsystem Message Handling Unit.

Type: [IoTSS_MessageHandlingUnit](#).

fvp_mps2.sse200.mhu1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram0.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram1.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram2.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3

IoT Subsystem Memory Protection Controller.

Type: [IoTSS_MemoryProtectionController](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.mpc_iotss_internal_sram3.gating_disabled_thread_event

A SchedulerThreadEvent instance is an auto-reset boolean condition variable other threads can wait on.

Type: [SchedulerThreadEvent](#).

fvp_mps2.sse200.nmi_or_gate

Or Gate.

Type: [OrGate](#).

fvp_mps2.sse200.nonsecure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.nonsecure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram0_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram0_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram1_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram1_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram2_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram2_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.ram3_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.ram3_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.s32k_timer.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.s32k_timer.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.s32k_timer.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.s32k_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.s32k_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block

MPS2 Secure Control Register Block.

Type: [MPS2_SecureCtrl](#).

fvp_mps2.sse200.secure_control_register_block.bus_mapper

Type: [PVBusMapper](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_iod

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_ns_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_iod

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.busslave_s_mps2

Type: [PVBusSlave](#).

fvp_mps2.sse200.secure_control_register_block.idau_busmaster

Type: [PVBusMaster](#).

fvp_mps2.sse200.secure_watchdog

ARM Watchdog Module.

Type: [CMSDK_Watchdog](#).

fvp_mps2.sse200.secure_watchdog.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.signal_router

Signal router.

Type: [SignalRouter](#).

fvp_mps2.sse200.sys_ppu

ARM Power Policy Unit (PPU) architectural model.

Type: [PPUv0](#).

fvp_mps2.sse200.sys_ppu.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer0.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer0.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.sse200.timer1

ARM Timer Module.

Type: [CMSDK_Timer](#).

fvp_mps2.sse200.timer1.busslave

Type: [PVBusSlave](#).

fvp_mps2.sse200.timer1.clk_div

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new

ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.sse200.timer1.counter

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.ssram1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.ssram2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.ssram2.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub0.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2c1

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2c1.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_i2s

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_i2s.bus_slave

Type: [PVBUSSlave](#).

fvp_mps2.stub_spi0

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi0.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.stub_spi2

RAM device, can be dynamic or static ram.

Type: [RAMDevice](#).

fvp_mps2.stub_spi2.bus_slave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer

Type: [SVOS_DualTimer](#).

fvp_mps2.svos_dualtimer.busmaster0

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster1

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster2

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busmaster3

Type: [PVBusMaster](#).

fvp_mps2.svos_dualtimer.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0

ARM Dual-Timer Module.

Type: [CMSDK_DualTimer](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter0

Internal component used by SP804 Timer module.

Type: [CounterModule](#).

fvp_mps2.svos_dualtimer.svos_dualtimer0.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer1.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer1.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer2.busslave

Type: PVBusSlave.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: ClockDivider.

fvp_mps2.svos_dualtimer.svos_dualtimer2.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer2.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3

ARM Dual-Timer Module.

Type: CMSDK_DualTimer.

fvp_mps2.svos_dualtimer.svos_dualtimer3.busslave

Type: [PVBusSlave](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div0

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.clk_div1

A ClockDivider is a library component that takes a ClockSignal on its input port (which could come from the output of a MasterClock, or from another ClockDivider), and generates a new ClockSignal on its output port, representing a clock frequency that is related to the input clock by the ratio of the multiply and divide parameters.

Type: [ClockDivider](#).

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter0

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.svos_dualtimer.svos_dualtimer3.counter1

Internal component used by SP804 Timer module.

Type: CounterModule.

fvp_mps2.switch_PSRAM_M7

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_PSRAM_M7.mapper

Type: [PVBusMapper](#).

fvp_mps2.switch_svos_dualtimer

Allow transactions to be routed arbitrarily.

Type: [PVBusRouter](#).

fvp_mps2.switch_svos_dualtimer.mapper

Type: [PVBusMapper](#).

fvp_mps2.telnetterminal0

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal1

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.telnetterminal2

Telnet terminal interface.

Type: [TelnetTerminal](#).

fvp_mps2.touchscreen_interface

MPS2 Touch Screen.

Type: [MPS2_TouchScreen](#).

fvp_mps2.touchscreen_interface.pvbusslave

Type: [PVBusSlave](#).

fvp_mps2.uart_overflows_or_gate

Or Gate.

Type: [OrGate](#).

idau.bus_bridge

Bridge incoming transactions to a PVDevice port.

Type: [PVBusBridge](#).

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is Final, that is for a developed product.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
1126-00	19 June 2024	Non-Confidential	Update for v11.26.
1125-00	13 March 2024	Non-Confidential	Update for v11.25.
1124-00	6 December 2023	Non-Confidential	Update for v11.24.
1123-00	13 September 2023	Non-Confidential	Update for v11.23.
1122-00	14 June 2023	Non-Confidential	Update for v11.22.
1121-00	22 March 2023	Non-Confidential	Update for v11.21.

Issue	Date	Confidentiality	Change
1120-00	7 December 2022	Non-Confidential	Update for v11.20.
1119-00	14 September 2022	Non-Confidential	Update for v11.19.
1118-01	26 August 2022	Non-Confidential	Update for Arm Corstone SSE-310 FVP.
1118-00	15 June 2022	Non-Confidential	Update for v11.18.
1117-00	16 February 2022	Non-Confidential	Update for v11.17.
1116-00	6 October 2021	Non-Confidential	Update for v11.16.
1115-00	29 June 2021	Non-Confidential	Update for v11.15.
1114-01	14 April 2021	Non-Confidential	Update for FVP_Base_AEMv8R.
1114-00	17 March 2021	Non-Confidential	Update for v11.14.
1113-00	9 December 2020	Non-Confidential	Update for v11.13.
1112-00	22 September 2020	Non-Confidential	Update for v11.12.
1111-00	9 June 2020	Non-Confidential	Update for v11.11.
1110-00	12 March 2020	Non-Confidential	Update for v11.10.
1109-00	28 November 2019	Non-Confidential	Update for v11.9.
1108-01	3 October 2019	Non-Confidential	Update for v11.8.1.
1108-00	5 September 2019	Non-Confidential	Update for v11.8.
1107-00	17 May 2019	Non-Confidential	Update for v11.7.

Issue	Date	Confidentiality	Change
1106-00	26 February 2019	Non-Confidential	Update for v11.6.
1105-00	23 November 2018	Non-Confidential	Update for v11.5.
1104-01	17 August 2018	Non-Confidential	Update for v11.4.2.
1104-00	22 June 2018	Non-Confidential	Update for v11.4.
1103-00	23 February 2018	Non-Confidential	Update for v11.3.
1102-00	17 November 2017	Non-Confidential	Update for v11.2.
1101-00	31 August 2017	Non-Confidential	Update for v11.1.
1100-00	31 May 2017	Non-Confidential	Update for v11.0. Document numbering scheme has changed.

For technical changes to this documentation, see the [Fast Models Release Notes](#).

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.

Convention	Use
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .



We recommend the following. If you do not follow these recommendations your system might not work.



Your system requires the following. If you do not follow these requirements your system will not work.



You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.



This information is important and needs your attention.



This information might help you perform a task in an easier, better, or faster way.



This information reminds you of something important relating to the current content.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Table 1: Arm publications

Document name	Document ID	Licensee only
Arm® Architecture Models	-	No
Arm® Corstone™ SSE-300 Example Subsystem Technical Reference Manual	101773	No
Arm® Corstone 1000	-	No
Arm® Corstone™-1000 Technical Overview	102360	No
Arm® Development Platforms wiki	-	No
Arm® Development Studio Getting Started Guide	101469	No
Arm® Development Studio User Guide	101470	No
Arm® MPS3 FPGA Prototyping Board Technical Reference Manual	100765	No
Arm® Neoverse™ v1 reference design Software Developer Guide	PJDOC-1779577084-33214	No
Corstone™-300	-	No
Fast Models Reference Guide	100964	No
Fast Models Tools User Guide	109415	No
Fast Models User Guide	100965	No
Getting started with your FVP	-	No
Iris Python Debug Scripting User Guide	101421	No
Iris User Guide	101196	No
Juno r2 Arm® Development Platform SoC Technical Reference Manual	DDI 0515	No
Neoverse™ Reference Design	-	No

Table 2: Arm publications

Document name	Document ID	Licensee only
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	No
Semihosting for AArch32 and AArch64 specification	-	No